Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Глибинна нейронна мережа для створення 3D моделей на основі зображень об'єкта
(тема)

Виконав:
студент 2 курсу, групи СШІм-21-2
Петрикін М.Ю.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту
(повна назва спеціалізації)

Керівник проф. Рябова Н.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ В.О. Філатов
(підпис) (прізвище, ініціали)

2023 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту (СШІ)
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Петрикіну Михайлу Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Глибинна нейронна мережа для створення 3D моделей на основі зображень об'єкта

затверджена наказом університету від 31 березня 20 23 р. № 306Ст
2. Термін подання студентом роботи до екзаменаційної комісії 18 травня 20 23 р.
3. Вихідні дані до роботи Науково-технічні публікації, дані Інтернет-джерел з інформацією про проведені дослідження у галузі створення 3D моделей з зображень або відео об'єктів

4. Перелік питань, що потрібно опрацювати в роботі
1) Аналіз предметної галузі
2) Методи та алгоритми глибинного навчання для перетворення 2D зображень у 3D моделі
3) Проведення експериментального дослідження
4) Аналіз результатів проведеного дослідження

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри)_____

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз предметної галузі	05.04.2023	виконано
2	Аналіз аналогів 3D моделювання	10.04.2023	виконано
3	Аналіз моделей глибинного навчання	17.04.2023	виконано
4	Експериментальне дослідження моделей	23.04.2023	виконано
5	Написання пояснювальної записки	28.04.2023	виконано
6	Попередній захист	13.05.2023	виконано
7	Захист перед ЕК	18.05.2023	

Дата видачі завдання 3 квітня 2023 р.

Студент _____
(підпис)

Керівник роботи _____ _____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 62 с., 25 рис., 2 табл., 1 дод., 22 джерела.

ГЕНЕРАТИВНА ЗМАГАЛЬНА НЕЙРОННА МЕРЕЖА, ГЛИБИННЕ НАВЧАННЯ, ЗАЛИШКОВА НЕЙРОННА МЕРЕЖА, МЕТОДИ 3D МОДЕЛЮВАННЯ, НЕЙРОННІ МЕРЕЖІ, ОБРОБКА ЗОБРАЖЕНЬ, ФОТОГРАММЕТРІЯ

Об’єкт дослідження – архітектура та навчання глибинних штучних нейронних мереж.

Предмет дослідження – методи моделювання генеративних штучних нейронних мереж та алгоритми їх навчання.

Мета роботи – моделювання 3D моделей з зображень об’єкта.

Методи дослідження – аналіз існуючих рішень з використанням нейронних мереж для моделювання зображень, розробка архітектури та алгоритму навчання генеративно змагальної глибинної нейронної мережі, програмна реалізація та графічне моделювання результатів.

Під час виконання кваліфікаційної роботи проведений теоретичний аналіз літературних джерел щодо методів моделювання глибинних штучних нейронних мереж та алгоритмів їх навчання в задачі генерації 3D моделей з використанням 2D зображень, наукових публікацій щодо розробки генеративно змагальних нейронних мереж для безпосередньої роботи з 2D зображеннями. Було виділено основні недоліки існуючих підходів та запропоновано архітектуру – генеративно змагальна нейронна мережа з використанням залишкової нейронної мережі.

ABSTRACT

Explanatory note: 62 p., 25 fig., 2 tables, 1 ann., 22 sources.

DEEP LEARNING, GENERATIVE COMPETITIVE NEURAL NETWORK, IMAGE PROCESSING, NEURAL NETWORKS, PHOTOGRAMMETRY, RESIDUAL NEURAL NETWORK, 3D MODELING METHODS

Object of research – architecture and learning of deep artificial neural networks.

The subject of research – modeling methods of generative artificial neural networks and their learning algorithms.

Purpose – to apply deep artificial neural network for create simulation of 3D models from object images.

Research methods – analysis of existing solutions using neural networks for image modeling, development of the architecture and learning algorithm of a generatively competitive deep neural network, software implementation and graphical modeling of results.

During the performance of the qualification work, a theoretical analysis of literary sources on modeling methods of deep artificial neural networks and algorithms for their training in the task of generating 3D models using 2D images, scientific publications on the development of generative competitive neural networks for direct work with 2D images was carried out. The main shortcomings of the existing approaches were highlighted and an architecture was proposed – a generative competitive neural network using a residual neural network.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

2D – двовимірне;

3D – тривимірне;

3D-R2N2 – тривимірної нейронна мережа рекурентної реконструкції;

CNN – Convolutional Neural Network – згорткова нейронна мережа;

DCNN – Deep Convolutional Neural Network – глибинна згорткова нейронна мережа;

DISN – Deep Implicit Surface Network – глибока неявна поверхнева мережа;

GAN – Generative Adversarial Network – генеративна змагальна мережа;

LSTM – Long Short-Term Memory – довга короткочасна пам’ять;

NeRF – Neural Radiance Field – поле нейронного випромінювання;

R-CNN – Region-based Convolutional Neural Network – регіональна згорточна нейронна мережа;

ReLU – Rectified Linear Unit – випрямлений лінійний блок;

ResNet – Residual Network – залишкова мережа.

ВСТУП

Зростання штучного інтелекту і технологій машинного навчання означає, що штучні мережі стали дедалі більш розвинутими і потужними. Розробники тепер можуть створювати штучні мережі, які можуть виконувати складні завдання, включаючи розпізнавання облич, голосові інтерфейси, машинне перекладання та багато іншого. Це значно збільшує можливості автоматизації, а також дозволяє розробникам створювати нові продукти та послуги, які були раніше неможливі.

Одна з ключових ролей штучних мереж у сучасному житті – це покращення продуктивності та ефективності роботи в бізнесі та промисловості. Штучні мережі дозволяють автоматизувати процеси, що раніше вимагали багато ручної праці. Наприклад, вони можуть бути використані для виявлення аномальної поведінки в системах безпеки, вирішення складних проблем зі зберіганням даних та покращенням ефективності виробничих процесів [1]. Штучні мережі також використовуються в медицині, де вони допомагають забезпечувати більш точні діагнози та зменшувати ризик помилок. Крім того, штучні мережі можуть бути використані для розв'язання складних задач у роботі з зображеннями.

Нові технології штучного інтелекту та машинного навчання дозволяють створювати більш складні та точні алгоритми комп’ютерного зору, що допомагає розв’язувати багато проблем у різних сферах. Наприклад, штучні мережі використовуються для візуалізації тривимірних об’єктів, що дозволяє зробити більш точні вимірювання та моделювання. Вони також можуть бути використані в будівництві та архітектурі, щоб створювати детальні плани та візуалізації проектів. Більш точна робота з тривимірними об'єктами також може знайти застосування у розробці віртуальної реальності та відеоігор. Одним із найважливіших використань штучних мереж є можливість покращення зображень від різних джерел, що

допомагає зробити їх більш чіткими та детальними, що знадобиться у медицині, науці, а також у бізнесі та маркетингу.

Світ навколо нас є тривимірним, тому робота над алгоритмами, які допомагають комп'ютерам зрозуміти цю притаманну цьому світу тривимірну структуру, є важливою сферою досліджень комп'ютерного зору. Створення 3D форми з 2D зображення завжди були важливим напрямком досліджень у цей області.

3D об'єкти можуть бути представлені багатьма способами. Використовуються вокселі та хмари точок для представлення тривимірних структур [2]. Воксельне представлення концептуально просте, але потребує високу резолюцію для захоплення тонких структур, а масштабування до цієї високої резолюції є нетривіальним. Хмари точок можуть зображати об'єкти без величезної кількості вершин, але вони недосконало відображають поверхню об'єкта. Крім того, вилучення з них об'єкти для візуалізації чи інших задач потребують постобробки. Натомість полігональна сітка (mesh) може чітко визначати тривимірні форми та є стандартним представленням, що використовуються в графічних програмах.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ФОРМАЛІЗОВАНА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз предметної області

Фотограмметрія – це наука про отримання достовірної інформації навколишнього середовища і фізичних об’єктів та за допомогою процесу запису, вимірювання та інтерпретації фотографічних зображень і моделей електромагнітного випромінювання та інших явищ [3]. Існує багато варіантів фотограмметрії. Одним із прикладів є вилучення тривимірних вимірювань із двовимірних даних, тобто зображень. Наприклад, відстань між двома точками, які лежать на площині або паралельній площині фотографічного зображення, можна розрахувати, вимірявши їх відстань на зображенні, якщо відомий масштаб зображення.

Зйомка цифрових зображень і фотограмметрична обробка включає кілька чітко визначених етапів, які дозволяють генерувати 2D або 3D цифрові моделі об’єкта як кінцевий продукт. Модель даних на рисунку 1.1 показує, який тип інформації може надходити та виходити з фотограмметричних методів. Тривимірні координати визначають розташування точок об’єктів у тривимірному просторі. Координати зображення визначають розташування об’єкта на плівці або електронному пристрої формування зображення. Просторова орієнтація [4] камери визначає розташування камери в просторі та куди вона направлена. Внутрішня орієнтація, яка характеризується такими параметрами, як фокусна відстань, дисторсія об’єктиву, визначає геометричні параметри процесу формування зображення.

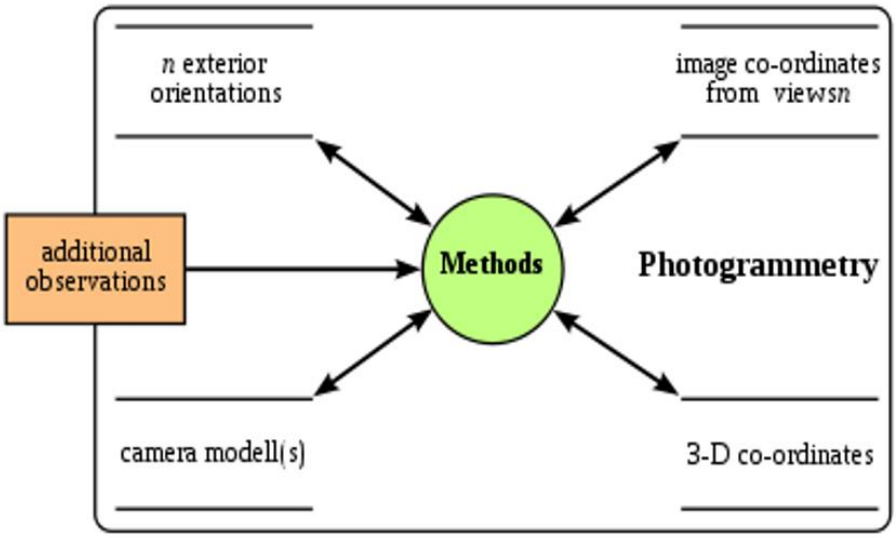

Рисунок 1.1 – Методи фотограмметрії

Стереофотограмметрія – окремий вид фотограмметрії, що визначає тривимірні координати об'єкта за допомогою вимірювань, зроблених на двох або більше фотографічних зображеннях, зроблених з різних позицій [5].

Фотограмметрія використовується в різних галузях, таких як медицина, інженерія, картографування, архітектура, виробництво, поліцейські розслідування, культурна спадщина та геологія. Археологи використовують її для швидкого створення планів великих або складних місць, а метеорологи використовують її для визначення швидкості вітру торнадо, коли неможливо отримати об’єктивні дані про погоду.

Перша галузь, яка адаптувала або використовувала цю технологію, це інженерія. При будівництві складної конструкції дуже важливо мати точні вимірювання. Використовуючи 3D-моделі, згенеровані за допомогою зображень з дронів, інженери можуть належним чином оцінити будівельні майданчики, що гарантує, що все проходить гладко, без будь-яких перешкод.

Фотограмметрія також відіграє важливу роль у галузі кіно та розваг. Протягом багатьох років кінематографісти почали значною мірою

покладатися на фотограмметрію для 3D-моделювання та точних вимірювань, необхідних для побудови світу для ігор і фільмів.

1.2 Проблематика створення 3D моделей

Швидке та автоматичне 3D-прототипування об’єктів стало інновацією, що змінила правила гри в багатьох сферах, пов'язаних з електронною комерцією, візуалізацією та архітектурою, серед яких можна назвати декілька. Ця тенденція посилилася тепер, коли 3D-друк став демократичною технологією, а методи отримання 3D-зображень демократизованою технологією, а методи отримання 3D-зображень стали точними та ефективними. Крім того, ця тенденція також пов’язана з поширенням великомасштабних репозиторіїв 3D-моделей об’єктів, таких як ShapeNet.

Однак більшість сучасних методів реконструкції 3D-об’єктів мають ряд обмежень. Деякі обмеження полягають у тому, що:

– об’єкти повинні спостерігатися з великої кількості точок зору. Або, що еквівалентно, точки зору повинні мати відносно невелику базову лінію. Це проблема, коли користувачі хочуть реконструювати об’єкт за кількома видами або, в ідеалі, лише за одним видом;

– зовнішній вигляд об’єктів (або їхні функції відбиття) вважатимуться ламбертіанськими (тобто такими, що не відбивають), а альбедо – неоднорідними (тобто такими, що містять багато неоднорідностей), багатими на неоднорідні текстури).

Ці обмеження випливають з низки ключових технічних припущень. Одне з них типовим припущенням є те, що ознаки можуть бути узгоджені між поглядами, як це передбачається більшістю методів, заснованих на SFM або SLAM. Якщо точки зору розділені великою базовою лінією, встановлення відповідностей ознак є надзвичайно проблематичним через локальні зміни зовнішнього вигляду або самообмеження. Крім того,

відсутність текстури на об’єктах та віддзеркалення також дуже ускладнюють проблему зіставлення ознак.

1.3 Актуальність перетворення 2D зображень до 3D моделей

Зв’язок між 2D зображеннями та 3D моделями дуже важливий для багатьох галузей, таких як комп’ютерна графіка, візуалізація даних, медицина, архітектура та інтерактивний дизайн.

Перетворення 2D зображень у 3D моделі відкриває безліч можливостей для більш точної візуалізації об’єктів та предметів у тривимірному просторі. Це дозволяє досліджувати та аналізувати їх з більшою точністю та глибиною, що допомагає у покращенні проектування, планування та вирішенні проблем.

Основні переваги перетворення 2D зображень у 3D моделі:

– реалістичність візуальних ефектів. У комп’ютерній графіці такі моделі дозволяють створювати більш реалістичні візуальні ефекти, які можуть бути використані у фільмах, відеоіграх, віртуальних турах та інтерактивних додатках;

– покращення точності та глибини аналізу. У медицині, такі моделі можуть бути використані для точнішого діагностування та планування хірургічних втручань. Крім того, перетворення 2D зображень у 3D моделі дозволяє вирішувати питання просторової орієнтації та взаємодії об'єктів. Це допомагає у створенні більш ефективних дизайнів, управлінні процесами виробництва та підвищенні якості продукту;

– візуалізація даних. У візуалізації даних, перетворення 2D зображень у 3D моделі може бути використано для створення графіків та діаграм, які дозволяють краще розуміти складні дані та їх зв’язки у тривимірному просторі. Це може бути особливо корисним у наукових дослідженнях, інженерії, фінансах та багатьох інших галузях, де точна візуалізація даних є критично важливою;

– планування та дизайн. У архітектурному дизайні, перетворення 2D зображень у 3D моделі дозволяє проектувати будівлі та інтер'єри з більшою точністю та реалістичністю, що допомагає у покращенні якості та ефективності проектів. Крім того, такі моделі можуть бути використані для віртуальних туров, які допомагають клієнтам краще зрозуміти проект перед його реалізацією;

– інтерактивний дизайн. У розвитку інтерактивних додатків, таких як ігри та віртуальна реальність, перетворення 2D зображень у 3D моделі дозволяє створювати більш ефективну та реалістичну взаємодію між користувачем та додатком.

Отже, перетворення 2D зображень у 3D моделі має безліч застосувань та може бути корисним для багатьох галузей. Воно допомагає покращити точність та глибину аналізу, забезпечує реалістичність візуальних ефектів, покращує візуалізацію даних, сприяє кращому плануванню та дизайну, а також допомагає у створенні більш ефективних та реалістичних інтерактивних додатків.

1.4 Аналіз існуючих рішень

Команда дослідників із NVIDIA та Стенфордського університету представила EG3D, нову гібридну мережеву архітектуру, яка може генерувати 2D-зображення людських і котячих обличь із високою резолюцією у режимі реального часу та генерувати 3D-об’єкти на основі цих зображень [6]. Мета проекту полягала в тому, щоб підвищити обчислювальну ефективність і якість зображення 3D GAN, не покладаючись надмірно на апроксимації, які впливають на узгодженість кількох ракурсів і якість форми.

Також NVIDIA створила нову технологію, покращену версію нейронного поля випромінювання яскравості (neural radiance field, NeRF). Ця технологія, яка отримала назву Instant NeRF, є найшвидшою технікою

NeRF на сьогоднішній день, яка в деяких випадках швидша більш ніж у 1000 разів [7]. Моделі потрібні лише секунди, щоб потренуватися на кількох десятках фотознімків, та дані про кути камери, з яких вони були зроблені, щоб відтворити отриману 3D-сцену протягом десятків мілісекунд. Instant NeRF скорочує час рендерингу у декілька раз. Він базується на техніці, розробленій NVIDIA, яка називається кодування хеш-сітки з різною роздільною здатністю, яка оптимізована для ефективної роботи на графічних процесорах NVIDIA. Використовуючи новий метод кодування вхідних даних, можна досягати високоякісних результатів за допомогою крихітної нейронної мережі, яка швидко працює.

Попри те, що нова технологія від NVIDIA дозволяє швидко отримувати якісні 3D-сцени, вона все ж таки має недоліки. Використання цієї технології потребує неперервну зйомку об’єкта з різних ракурсів. Це вимагає велику концентрацію користувача, та робить цю технологію складною у використанні, коли потрібна швидка приблизна модель об’єкта.

Ще одним сервісом, який представляє послуги трансформування 2D зображень у 3D є Alpha3D. Alpha3D-це платформа на основі штучного інтелекту, яка дозволяє користувачам автоматично перетворювати 2D-зображення реальних продуктів у 3D-цифрові активи для AR, VR, ігор та метапростору. Великим мінусом являється не реалістичність 3D моделей у Alpha3D і наявність тільки платної підписки для створення 3D моделей.

Також існує багато програмних застосунків для більш ручного створення 3D моделей з 2D зображень, їх перечень можна побачити у таблиці 1.1.

Таблиця 1.1 – Застосунки для переведення 2D зображення у 3D

Застосунок	Вид	Платформа	Складність
Selva 3D	З 2D в 3D	Online	Легко
Smoothie 3D	З 2D в 3D	Online, iPad	Середньо
Vectary	Застосунок для	Online	Легко

Продовження таблиці 1.1

	3D моделювання		
Reliefmod	З 2D в 3D	Online	Легко
Meshroom	Застозунок для фотограметрії	Windows, Linux	Легко
Insight3d	Застозунок для фотограметрії	Windows, Linux	Середньо
Metashape	Застозунок для фотограметрії	Windows, Mac, Linux	Середньо

Selva 3D – це веб-сервіс, який може перетворювати зображення логотипів, креслень та інших базових дизайнів на 3D-моделі. Речі з чітко окресленими лініями та лише 1-3 кольорами найкраще підходять для цього типу сервісу. Це програмне забезпечення корисне тим, що в ньому можна швидко додати свій дизайн до таких предметів, як чохли для телефонів.

Smoothie 3D – це потужний 3D-інструмент для iPad або інтернету, який дозволяє створювати складні органічні 3D-моделі з 2D-зображень. Інструмент працює так: потрібно намалювати контур об’єкта, який необхідно перетворити на 3D для 3D-друку. Потім інструмент аналізує область в межах намальованої ділянки, щоб перетворити її на 3D-форму. Він також використовує колір і дизайн з фотографії, щоб надати 3D-моделі текстуру. Також в цьому застосунку можна дублювати частини, щоб створювати повністю симетричні органічні моделі. Наприклад, якщо ви моделюєте тварину, ви можете скористатися програмою, щоб намалювати одну сторону, а потім продублювати і віддзеркалити її, щоб автоматично створити іншу сторону.

Vectary – це онлайн 3D-пакет, призначений в основному для створення макетів і макетів продуктів, але він має деякі базові інструменти, які також використовуються для перетворення 2D-зображення в 3D-модель для 3D-друку. Хоча в ньому є бібліотека готових 3D-активів, можна швидко

відкрити порожній файл проекту, імпортувати своє зображення, видавити його або відредагувати іншим способом. Також в ньому можна імпортувати та експортувати файли на пристрій і з нього, що робить його чудовим варіантом для базового перетворення 2D в 3D.

Meshroom – це настільне програмне рішення для фотограмметрії з відкритим вихідним кодом для комп'ютерів під управлінням Windows і Linux. Щоб створити 3D-модель, потрібно завантажити кілька зображень об’єкта з різних ракурсів, а програма з’ясовує, як виглядає об’єкт, на основі посилань. Це досить потужне програмне забезпечення, тому воно вимагає певного часу на освоєння. Він використовує систему вузлів, подібну до Blender, яка може заплутати, якщо ви не звикли до неї. Вона дозволяє швидко додавати або змінювати атрибути об’єктів у сцені, тож це корисний робочий процес.

Insight3D – це безкоштовне програмне забезпечення для фотограмметрії з відкритим вихідним кодом, сумісне з Windows. Воно працює, витягуючи інформацію з метаданих камери для позиціонування зображень відносно напрямку, в якому вони були зроблені. Програма також відстежує точки інтересу (такі як текстура та особливості об’єкта), щоб визначити положення камери.

На додаток до інформації, взятої з самого зображення, також можна задати точки, від яких програма буде відштовхуватися. Також можливе вручну додати вершини, щоб показати, де починаються і закінчуються кути і ребра.

Ще одне програмне рішення для настільних комп’ютерів, Metashape, трохи зручніше за своїм дизайном, ніж Meshroom. Він доступний у стандартній та професійній версіях, причому стандартна версія – це урізана версія, а професійна – платний ресурс з повним доступом до програмного забезпечення.

Стандартна версія програми має менше функцій, ніж професійна, але між ними також існує досить значна різниця в ціні. Стандартна ліцензія дає

вам доступ до основних функцій, тоді як професійна ліцензія має більш розширені можливості.

Деякі з цих функцій включають автоматичне виявлення об’єктів, таких як хмари та лінії електропередач. Крім того, вона має вбудований скрипт Python для текстових команд, що може значно заощадити час для великих проектів.

1.5 Постановка задачі дослідження

Під час виконання дослідження необхідно аналізувати архітектури для створення 3D моделей з 2D зображень і розробити глибоку нейронну мережу, яка буде використовувати архітектуру GAN для генерації відповідних 3D моделей з вхідного 2D зображення. Для отримання високоякісних 3D моделей можна використовувати залишкову нейронну мережу як базову мережу для витягування ознак з вхідних зображень, отримання більш точних результатів та як рішення проблему згасаючого градієнту.

Навчання мережі повинне здійснюватися з використанням методів навчання без вчителя, що включають генеративну складову, що повинне забезпечувати створення високоякісних 3D моделей з вхідних зображень.

Після тренування мережі, необхідно перевірити її точність та ефективність на нових даних, які не входили до навчального набору даних. Також треба використовувати методи візуалізації, щоб відображати отримані 3D моделі та порівнювати їх зі вхідними 2D зображеннями.

2 МАТЕМАТИЧНИЙ ОГЛЯД МОДЕЛЕЙ

2.1 Штучні нейронні мережі

Штучна нейронна мережа – це математична модель, яка дозволяє моделювати поведінку нейронів мозку для вирішення різноманітних завдань штучного інтелекту. Вона складається з взаємопов'язаних нейронів, які передають інформацію через зв'язки (ваги) між ними.

Один нейрон можна представити в такій формі:

$$y = \sum_{i=0}^{n} w_i b_i + b, \tag{2.1}$$

де x_i – вхідний сигнал;

w_i – вага;

b – зсув;

f – функція активації;

y – вихідний сигнал нейрону.

Функція активації може бути будь-якою не лінійною функцією, наприклад, сигмоїдальною або ReLU (Rectified Linear Unit).

Багатошарова нейронна мережа (multilayer neural network) – це нейронна мережа, яка містить кілька шарів нейронів. Кожен шар складається з вузлів (нейронів), які приймають сигнали від попереднього шару і передають їх на наступний шар.

Найпоширенішою формою багатошарових нейронних мереж є так звана мережа прямого поширення, у якій сигнали поширюються лише в одному напрямку, від вхідного до вихідного шарів, без зворотного зв'язку. Така мережа складається з трьох типів шарів:

– вхідний шар (input layer) – це шар, до якого надходять дані від зовнішнього середовища. Кожен вузол вхідного шару представляє один вхідний параметр;

– приховані шари (hidden layers) – це шари, які не мають прямого зв’язку з вхідними або вихідними даними. Кожен вузол прихованого шару обчислює зважену суму вхідних сигналів з попереднього шару, застосовуючи до них функцію активації;

– вихідний шар (output layer) – це шар, з якого отримуються відповіді від мережі. Кожен вузол вихідного шару виконує остаточну операцію обчислення значення вихідного параметру на основі обчислень в прихованих шарах.

Прикладом багатошарової нейронної мережі може бути перцептрон з двома прихованими шарами. У такому випадку на вхід мережі подається вектор даних, який пройшов через перший шар нейронів, потім через другий, і так далі до останнього шару, який називається вихідним шаром. Кожен нейрон в кожному шарі отримує вхідні дані з попереднього шару, виконує певні обчислення з використанням ваг, які відповідають за взаємозв'язки між нейронами, і передає результат в наступний шар.

Математично багатошарова нейронна мережа може бути представлена як функція з першого шару нейронів, яка виконує лінійну комбінацію вхідних даних з вагами, додає зміщення і передає результат через нелинійну функцію активації. Результат передається в наступний шар нейронів, де процес повторюється. Таким чином, багатошарова нейронна мережа виконує послідовні лінійні та нелинійні перетворення, що дозволяє їй розв’язувати більш складні задачі, ніж проста одношарова мережа.

2.2 Архітектура 3D-R2N2

Для того, щоб обійти проблеми які мають зв’язок з великими базовими лініями або неламбертіанськими поверхнями, популярними стали методи 3D об’ємної реконструкції, такі як вирізання простору та їх імовірнісні розширення. Ці методи припускають, що об’єкти точно

відокремлені від фону або що камери відкалібровані, що, звісно, не відповідає дійсності в багатьох застосуваннях.

Інше припущення полягають у наявності попередніх знань про зовнішній вигляд і форму об’єкта. Перевага використання попередніх знань полягає в тому, що подальший метод реконструкції менш залежний від пошуку точних відповідностей ознак. Таким чином, методи, що базуються на попередніх знаннях форми, можуть працювати з меншою кількістю зображень, приклад яких можна побачити на рисунку 2.1, і з меншою кількістю припущень про функцію відбиття об’єкта. Попередні форми зазвичай кодуються у вигляді 3D-примітивів, або запозичуються з багатих репозиторіїв 3D-моделей, наприклад CAD, завдяки чому концепція відображення зображень облич у вигляді 3D-моделей була досліджена в набагато більшому обсязі. Також були введені складні математичні формулювання для адаптації 3D-моделей форми моделей з різним ступенем спостереження та різними стратегіями регуляризації.

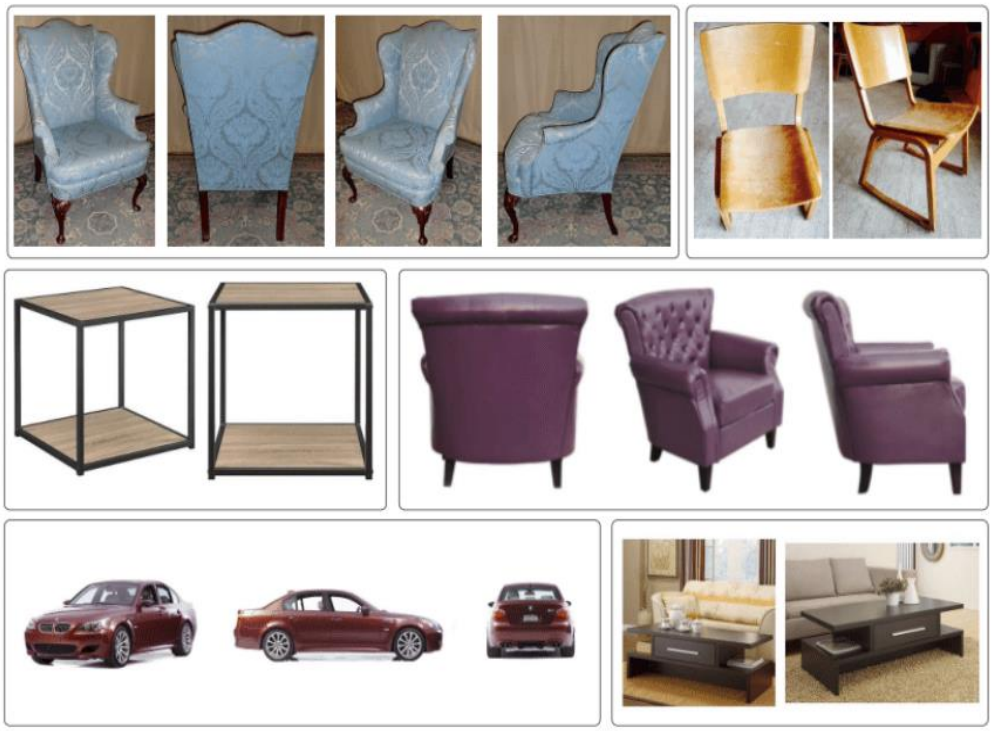

Рисунок 2.1 – Мінімальна кількість зображень для 3D реконструкції

Мережа 3D-R2N2 реалізує ідею , що замість того, щоб намагатися підібрати відповідну 3D-форму до спостереження об’єкта і адаптуватися до неї, можна використовувати глибокі згорткові нейронні мережі, щоб вивчити відображення спостережень на 3D-форми об’єктів, що лежать в їх основі [8]. Мережа використовує Декодер і Енкодер, а також архітектуру LSTM [9], отримує одне або декілька зображень об’єкта з різних точок зору і виводить реконструкцію об’єкта у вигляді тривимірної сітки зайнятості. Також мережа не потребує жодних міток класів об’єктів чи анотацій зображень. Спрощену схему архітектури можна побачити на рисунку 2.2.

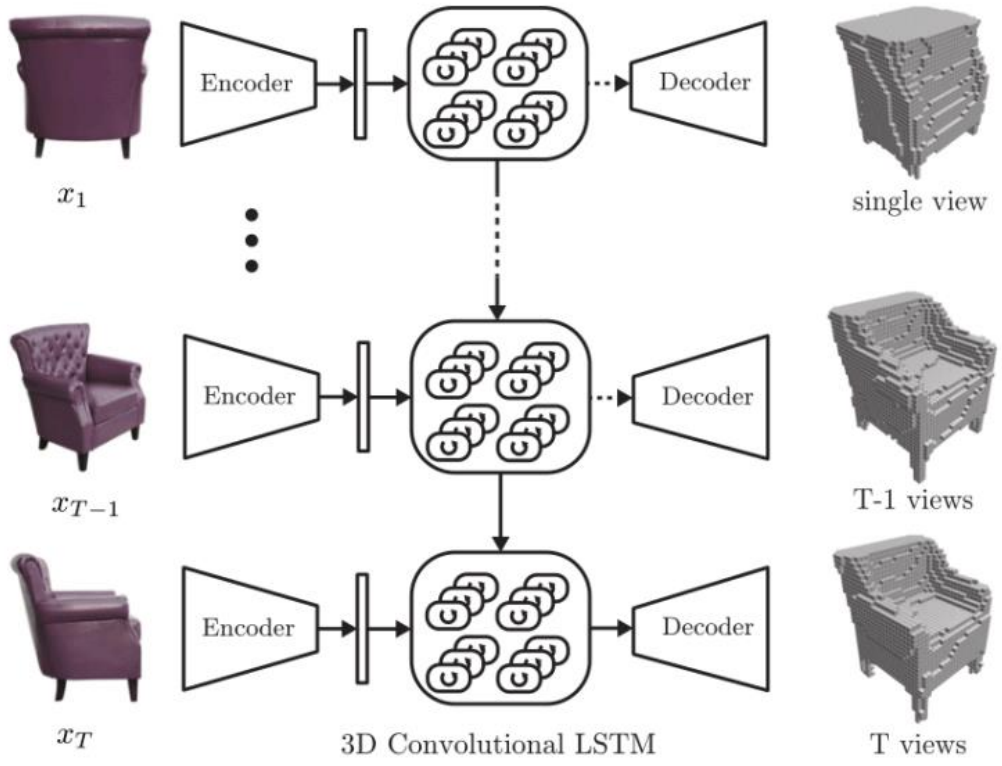

Рисунок 2.2 – Архітектура 3D-R2N2

Блок LSTM явно контролює потік від входу до виходу, що дозволяє мережі подолати проблему зникаючого градієнта. Зокрема, LSTM складається з чотирьох компонентів: блоків пам'яті (комірка пам'яті та прихований стан) і трьох вентилів, які керують потоком інформації від входу до прихованого стану (вхідний вентиль), від прихованого стану до

виходу (вихідний вентиль) і від попереднього прихованого стану до поточного прихованого стану (вентиль забування). Більш формально, на часовому кроці t, коли надходить новий вхідний сигнал x_t, роботу LSTM-блоку можна виразити як:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i), \tag{2.2}$$

$$f_t = \sigma\big(W_f x_t + U_f h_{t-1} + b_f\big), \tag{2.3}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_0), \tag{2.4}$$

$$s_t = f_t \cdot s_{t-1} + i_t \cdot \tanh\,(W_s x_t + U_s h_{t-1} + b_s), \tag{2.5}$$

$$h_t = o_t \cdot \tanh\,(s_t), \tag{2.6}$$

де i_t – вхідний вентиль;

f_t – вихідний вентиль;

o_t – вентиль забуття;

s_t та h_t – комірка пам’яті;

h_t – прихований стан відповідно;

символ · використовується для позначення поелементного множення;

t індекс – позначення активації в момент часу t;

W, U – матриці, які перетворюють поточний вхід x_t та попередній прихований стан h_{t-1} відповідно;

b – вектор зсувів.

Різновидом блоку LSTM є Gated Recurrent Unit (GRU) [10]. Перевагою GRU є те, що що порівняно зі стандартним LSTM потрібно менше обчислень. У GRU, вентиль оновлення керує як входом, так і вентилем забування. Інша відмінність полягає в тому, що що перед нелінійним перетворенням застосовується вентиль скидання:

$$u_t = \sigma(W_u \tau x_t + U_u * h_{t-1} + b_f), \quad (2.7)$$

$$r_t = \sigma(W_i \tau x_t + U_i * h_{t-1} + b_i), \quad (2.8)$$

$$h_t = (1 - u_t) \cdot h_{t-1} + u_t \cdot \tanh\,(W_h x_t + U_h(r_t \cdot h_{t-1}) + b_n), \quad (2.9)$$

де u_t – вентиль оновлення;

r_t – вентиль скидання;

h_t – прихований стан.

Ми використовуємо ті ж самі позначення, що і в LSTM для матриць та зміщень.

Мета мережі 3D-R2N2 полягає у виконанні одно та багаторекурсивних 3D-реконструкцій. Основна ідея полягає в тому, щоб використати можливості LSTM для збереження попередніх спостережень і поступового вдосконалення вихідної реконструкції, коли стає доступною більша кількість спостережень.

Кожна частина 3D-R2N2 має в собі кодер, рекурентний блок та декодер. Після кожного шару згортки розміщується LeakyReLU. Кодер перетворює RGB – зображення 127 × 127 у низькорозмірні дані, які потім подаються на 3D-LSTM. Потім декодер бере приховані стани 3D-LSTM і перетворює їх на остаточну карту зайнятості вокселів. Після кожного шару згортки створюється шар LeakyReLU.

Мережа 3D-R2N2 складається з трьох компонентів:

– 2D-згорткової нейронної мережі;

– архітектури під назвою 3D-згорткова LSTM;

– 3D-деконволюційної нейронної мережі [11].

Маючи одне або декілька зображень об’єкта, 2D-CNN спочатку кодує кожне вхідне зображення у низьковимірні ознаки. Потім, враховуючи закодовані вхідні дані, набір нещодавно запропонованих блоків 3D-LSTM або вибірково оновлює стани своїх комірок, або зберігає їх, закриваючи

вхідний вентиль. Нарешті, 3D-DCNN декодує приховані стани блоків LSTM і генерує тривимірну імовірнісну реконструкцію вокселів.

Основна перевага використання мережі на основі LSTM полягає в її здібності ефективно обробляти самовиключення об'єктів, коли в мережу подається кілька зображень. Мережа вибірково оновлює блоки пам’яті, які відповідають видимим частинам об’єкта. Якщо наступний вид показує частини, які раніше були самозакриті і не відповідають прогнозу, мережа оновить стани LSTM для раніше закритих ділянок, але збереже стани інших частин.

Також використовується Convolutional Neural Network (CNN) [12] для кодування зображень. Мережа складається зі стандартних шарів згортки, шарів об’єднання та негерметичних випрямлених лінійних блоків, за якими слідує повністю з’єднаний шар. Додавання залишкових зв’язків між стандартними шарами згортки ефективно покращує і прискорює процес оптимізації для дуже глибоких мереж.

На кожному часовому кроці кожен елемент у 3D-LSTM отримує однаковий вектор ознак від кодера, а також приховані стани від своїх сусідів за допомогою згортки $3 \times 3 \times 3$ ($W_s * h_{t-1}$) як вхідні дані, представлено на рисунку 2.3. 3D-LSTM реалізується як 3D Gated Recurrent Units, що представлено на рисунку 2.4.

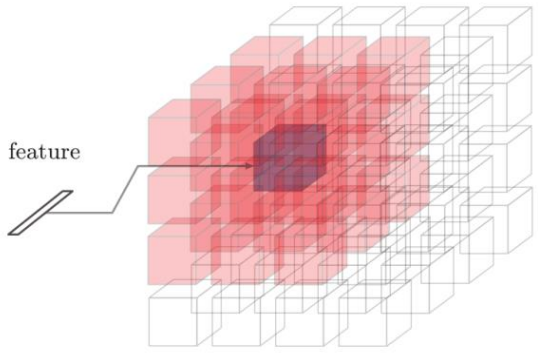

Рисунок 2.3 – Вхідні дані для кожного блоку LSTM

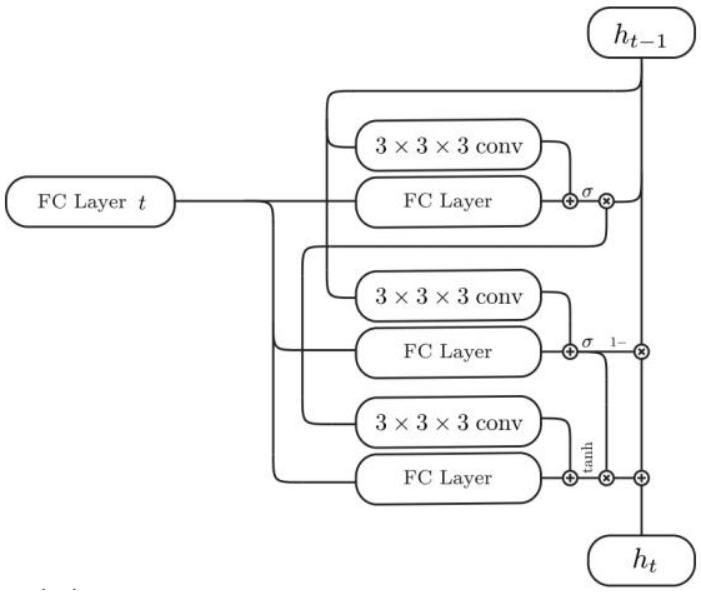

Рисунок 2.4 – 3D Convolutional GRUs

Основною частиною 3D-R2N2 є модуль рекурентності, який дозволяє мережі зберігати те, що вона бачила, і оновлювати пам’ять, коли вона бачить нове зображення. Наївним підходом було б використання звичайної LSTM-мережі. Рекурентний модуль на основі GRU можна виразити як:

$$u_t = \sigma(W_{fx}\tau(x_t) + U_f * h_{t-1} + b_f), \tag{2.10}$$

$$r_t = \sigma(W_{ix}\tau(x_t) + U_i * h_{t-1} + b_i), \tag{2.11}$$

$$h_t = (1 - u_t) \cdot h_{t-1} + u_t \cdot \tanh\,(W_h\tau(x_t) + \\ + U_h * (r_t \cdot h_{t-1}) + b_n). \tag{2.12}$$

Після отримання вхідної послідовності зображень x_1, x_2, … , x_n, 3D-LSTM передає прихований стан h_t на декодер, який збільшує роздільну здатність прихованого стану шляхом застосуванням 3D згорток, нелінійностей та 3D розбиття, поки не досягне цільової вихідної роздільної здатності.

Функція втрат мережі визначається як сума кросентропії по вокселях. Нехай кінцевий вихід у кожному вокселі (i, j, k) є розподілом Бернуллі [1 - p(i,j,k) , p(i,j,k)], де опускається залежність від входу X = {xt} t ∈ {1,...,T}, і нехай відповідне заповнення вокселів y(i,j,k) ∈ {0, 1}, тоді

$$L(X, y) = \sum_{i,j,k} y(i,j,k) \log\left(p_{(i,j,k)}\right) + \\ + \left(1 - y_{(i,j,k)}\right) \log\left(1 - p_{(i,j,k)}\right). \quad (2.13)$$

2.3 Архітектура DISN

Для подолання обмежень у вокселях, хмарах точок та сітках, існує альтернативне неявне представлення 3D-поверхонь, а саме, знакові функції відстаней (SDF).

Глибока неявня поверхнева мережу (Deep Implicit Surface Network, DISN) [13] використовується для прогнозування SDF на основі одноракурсних зображень, що забражено на рисунку 2.5.

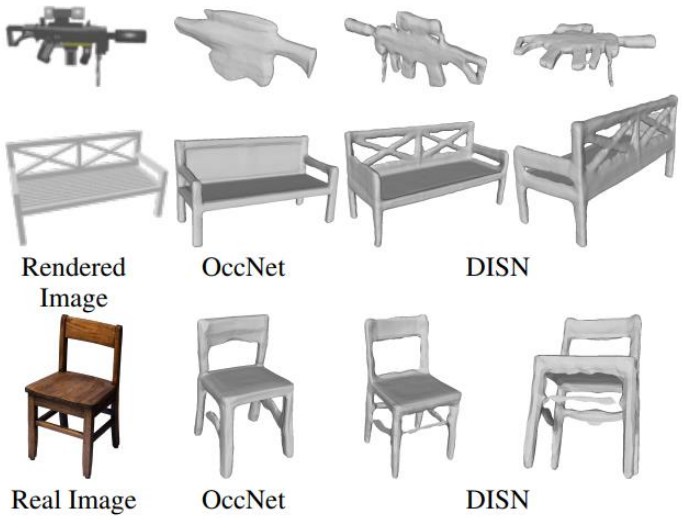

Рисунок 2.5 – Генерація моделей архітектурою DISN

SDF кодує підписану відстань кожної вибірки точок в 3D від межі базової фігури. Як показано на рисунку 2.6, за допомогою згорткової нейронної мережі, яка кодує вхідне зображення у вектор ознак, DISN прогнозує значення SDF заданої 3D точки, використовуючи цей вектор ознак.

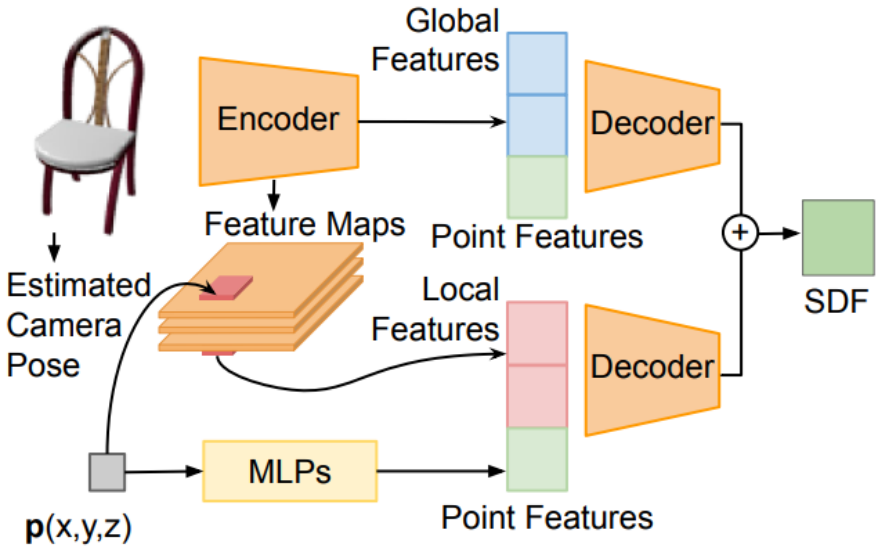

Рисунок 2.6 – Архітектура DISN

Шляхом вибірки різних розташувань 3D-точок DISN здатна генерувати неявне поле підстилаючої поверхні з нескінченною роздільною здатністю. Більше того, без необхідності припущення про фіксовану топологію, метою регресії для DISN є точна істина, а не наближена метрика.

В той час як багато методів одноракурсної 3D-реконструкції, які вивчають вбудовування фігури з 2D-зображення, здатні захопити глобальні властивості форми, вони мають тенденцію ігнорувати деталі такі як отвори або тонкі структури. Такі дрібнозернисті деталі займають лише невелику частину у 3D просторі і, таким чином, жертвуючи ними, не можна дізнатись про кількість великих втрат у порівнянні з реальною формою. Однак, такі результати можуть бути візуально незадовільними.

Щоб вирішити цю проблему, вводиться модуль виділення локальних особливостей. Зокрема, оцінюються параметри точки зору на вхідному зображенні. Через це можна спроектувати кожну точку запиту на вхідне зображення, щоб знайти відповідну локальну ділянку. Локальні особливості з таких ділянок виділяються і використовуються у поєднанні з глобальними характеристиками зображення для прогнозування значень SDF для 3D точок [14].

Цей модуль дозволяє мережі вивчати відношення між спроектованими пікселями та тривимірним простором і значно покращує якість реконструкції дрібнозернистих деталей у результуючій тривимірній формі. DISN здатна генерувати деталі форми, такі як візерунки на спинці лавки та отвори на руків'ї гвинтівки, які попередні сучасні методи не могли створити. DISN є першою моделлю глибокого навчання, яка здатна відтворювати такі високоякісні деталі з одноракурсних зображень.

Як показано на рисунку 2.7, SDF – це неперервна функція, яка відображає задану просторову точку $p = (x, y, z) \in \mathbb{R}^3$ у дійсне значення $s \in \mathbb{R}$: $s = SDF(p)$. Замість більш поширених 3D уявлень, таких як глибина, абсолютне значення s вказує на відстань від точки до поверхні, тоді як знак s показує, чи знаходиться точка всередині або зовні поверхні. Ізоповерхня $s_0 = \{ p \mid SDF(p) = 0 \}$ неявно представляє базову 3D-форму.

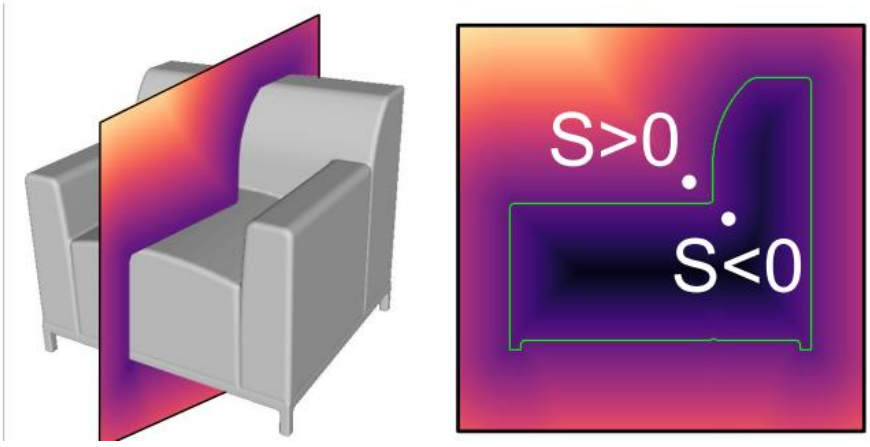

Рисунок 2.7 – Візуалізація SDF

2.4 Архітектура Mesh R-CNN

Архітектура Mesh R-CNN [15] була розроблена з метою створення системи, яка на вході отримує одне зображення, виявляє всі об’єкти і виводить мітку категорії, обмежувальну рамку, маску сегментації для кожного виявленого об’єкта, маску сегментації та тривимірну трикутну сітку для кожного об’єкта. Вихідні сітки мережі не повинні бути обмежені фіксованою топологією, щоб вміщувати широкий спектр складних об’єктів реального світу. Архітектуру можна побачити на рисунку 2.8.

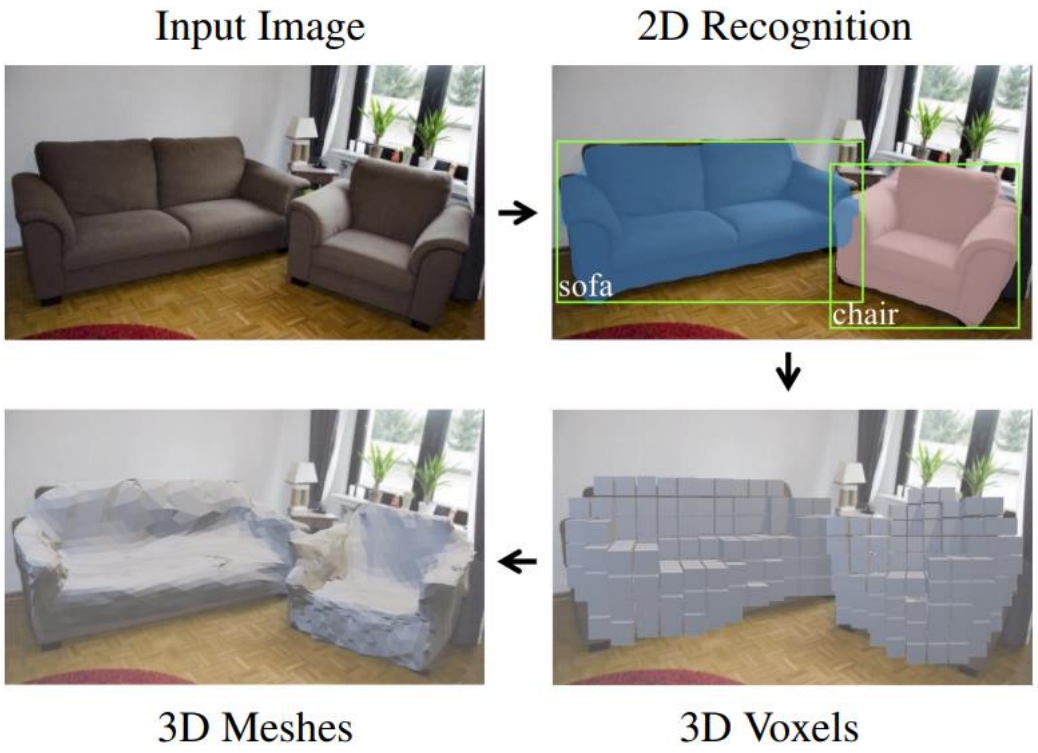

Рисунок 2.8 – Обробка зображення мережею Mesh R-CNN

Зокрема, мережа працює на основі Mask R-CNN [16], системи 2D-сприйняття. Mask R-CNN – це наскрізний регіональний детектор об'єктів. Він вводить одне RGB-зображення і виводить обмежувальну рамку, мітку категорії та маску сегментації для кожного виявленого об’єкта. Зображення спочатку проходить через магістральну мережу (наприклад, ResNet-50-

FPN). Далі мережа регіональних пропозицій (RPN) видає пропозиції щодо об’єктів, які обробляються блоками класифікації об’єктів і прогнозування маски.

Частково успіх Mask R-CNN завдячує RoIAlign [17], який виокремлює ознаки регіону з ознак зображення, зберігаючи при цьому вирівнювання між вхідним зображенням та ознаками, що використовуються у фінальному прогнозі.

Також 3D-форми виводяться за допомогою нового сіткового предиктора, що складається з воксельної гілки та гілки уточнення сітки. Блок вокселів спочатку оцінює грубу 3D вокселізацію об’єкта, яка перетворюється на початкову трикутну сітку. Блок уточнення сіті потім коригує положення вершин цієї початкової сіті за допомогою послідовності шарів згортки графів, що працюють над краями сіті.

Блок вокселів та блок уточнення мережі є гомологічними до існуючих блоків вікна та маски у Mask RCNN. Всі вони приймають на вхід вирівняні за зображенням ознаки, що відповідають пропозиціям RPN. Воксельні та сіткові втрати додаються до втрат у вікні та масці, і вся система навчається наскрізь. На виході ми отримується набір блоків разом з прогнозованими оцінками об’єктів, масками та 3D-формами.

В основі системи лежить предиктор сітки, який отримує згорнуті ознаки, вирівняні по обмежувальній рамці об’єкта, і прогнозує трикутну сітку, що дає повну тривимірну форму об’єкта. Як і в Mask R-CNN, підтримується відповідність між вхідним зображенням та ознаками, що використовуються на всіх етапах обробки, за допомогою операторів вирівнювання за областями та вершинами (RoIAlign та VertAlign). Метою є отримання специфічних для кожного екземпляра 3D-форм усіх об’єктів на зображенні. Таким чином, кожна прогнозована мережа повинна мати специфічну для даного екземпляра топологію (рід, кількість вершин, граней, з’єднаних компонентів) і геометрію (положення вершин).

Прогнозуються різні топології сітки, застосовуючи послідовність операцій виведення форми. По-перше, гілка вокселів робить вокселізовані передбачення форми кожного об’єкта, подібно до гілки маски R-CNN. Ці передбачення перетворюються на сітки і коригуються головкою уточнення сіток, що дає остаточні передбачені сітки.

Результатом роботи сіткового предиктора є трикутна сітка T = (V, F) для кожного об’єкта. V = {vi ∈ R 3} – множина позицій вершин а F ⊆V ×V ×V – множина трикутних граней.

Гілка вокселів прогнозує сітку ймовірностей заповнення вокселів, що дає курсову 3D-форму кожного виявленого об'єкта. Її можна розглядати як 3D-аналог гілки прогнозування маски R-CNN: замість того, щоб прогнозувати сітку M × M, яка дає форму об'єкта на площині зображення, прогнозує сітку G × G × G, яка дає повну 3D форму об’єкта.

Як і в Mask R-CNN, підтримується відповідність між вхідними ознаками та передбаченими вокселями, застосовуючи невелику повністю згорткову мережу до карти ознак, отриманої за допомогою RoIAlign. Ця мережа створює карту ознак з G каналами, що дає стовпчик оцінок зайнятості вокселів для кожної позиції і вхідних даних.

Підтримувати попіксельну відповідність між зображенням і прогнозами в 3D складно, оскільки об’єкти стають меншими, коли вони віддаляються від камери. Для вирішення цієї проблеми використовується матрицю камери для прогнозування фрустумоподібних вокселів.

Воксельна гілка створює 3D-сітку ймовірностей заповнення, що дає грубу форму об’єкта. Для того, щоб передбачити більш дрібнозернисті 3D форми, ці воксельні прогнози у трикутну сітку, яку можна передати до гілки уточнення мережі.

Ця прогалина заповнюється за допомогою операції, яка називається кубіфікація. Вона вводить ймовірності зайнятості вокселів та поріг для бінаризації зайнятості вокселів. Кожен зайнятий воксель замінюється сіткою кубічного трикутника з 8 вершинами, 18 ребрами та 12 гранями.

Спільні вершини та ребра між сусідніми зайнятими вокселями об’єднуються, а спільні внутрішні грані видаляються. В результаті отримується водонепроникна сітка, топологія якої залежить від прогнозів вокселів.

Кубіфікація має бути ефективною і серійною. Це не є тривіальним завданням. Альтернативно, маршові куби можуть виділити ізоповерхню з воксельної сітки, але це значно складніше.

Воксельна гілка навчається для мінімізації бінарної перехресної ентропії між прогнозованою зайнятістю вокселів ймовірністю заповнення вокселів та їх реальним заповненням.

Кубічна сіть з гілки вокселів забезпечує лише грубу 3D-форму, і вона не може точно моделювати тонкі структури, такі як ніжки стільців. Гілка уточнення сіті обробляє цю початкову кубічну сіть, уточнюючи положення її вершин за допомогою послідовності етапів уточнення. Аналогічно, кожен етап уточнення складається з трьох операцій: вирівнювання вершин, яке витягує особливості зображення для вершин; згортка графа, яка поширює інформацію вздовж ребер мережі та уточнення вершин, яке оновлює позиції вершин. Кожен шар мережі зберігає тривимірну позицію v_i та вектор ознак f_i для кожної вершини сітки.

На першому етапі гілки уточнення сітки VertAlign виводить початковий вектор ознак для кожної вершини. На наступних етапах вихідні дані VertAlign об’єднуються з характеристикою вершини з попереднього етапу.

Згортання графів поширює інформацію вздовж ребрами сітки. За вхідними характеристиками вершин обчислюються оновлені характеристики:

$$f_i' = ReLU(W_0 f_i + \sum_{j \in N(i)} W_1 f_i), \qquad (2.14)$$

де N(i) – кількість сусідів і-ї вершини у сітці;

W0 та W1 – вивчені вагові матриці.

На кожному етапі гілки уточнення сітки використовує декілька шарів згортки графа шарів згортки графів для агрегування інформації в локальних областях сітки.

Уточнення вершин обчислює оновлені позиції вершин

$$v_i' = v_i + \tanh\left(W_{vert}[f_i; v_i]\right), \tag{2.15}$$

де Wvert – вивчена вагова матриця.

Це оновлює геометрію сітки, її топологія залишається незмінною. Кожен етап гілки уточнення сіті завершується уточненням вершин, створюючи проміжний результат, який далі уточнюється на наступному етапі.

Втрати в сітці Визначення втрат, які діють безпосередньо на трикутних сітках, є складним завданням, тому використовується функція втрат, визначена на скінченній множині точок. Мережа представляється хмарою точок, щільно дискретизуючи її поверхню. Отже, втрати для хмари точок наближаються до втрат для фігур.

Також використовується операція диференційованої дискретизації сітки для рівномірної вибірки точок та їхніх нормальних векторів з поверхні сітки. Для цього реалізується ефективний дискретизатор пакетної вибірки. Ця операція використовується для вибірки хмари точок P з наземної істинної сітки та хмари точок P з кожного проміжного прогнозу сітки з нашої моделі.

Для двох хмар точок P, Q з нормальними векторами, нехай Λ P,Q = {(p, arg minq kp - qk) : p ∈ P} – множина пар (p, q) таких, що q є найближчим сусідом точки p у Q, і нехай up - одинична нормаль до точки p. Відстань фаски між хмарами точок P і Q задається формулою

$$L_{cham}(P,Q) = |P|^{-1} \sum_{(p,q)\in \wedge_{P,Q}} ||p-q||^2 + \\ + |Q|^{-1} \sum_{(q,p)\in \wedge_{Q,P}} ||q-p||^2. \quad (2.16)$$

Абсолютна нормальна відстань задається формулою

$$L_{norm}(P,Q) = -|P|^{-1} \sum_{(p,q)\in \wedge_{P,Q}} |u_p \cdot u_q| - |Q|^{-1} \sum_{(q,p)\in \wedge_{Q,P}} |u_q \cdot u_p|. \quad (2.17)$$

Абсолютна відстань по нормалі дорівнює відстані по фасці та нормалі. Втрати рахуються як штрафи за невідповідність позицій та нормалей між двома хмарами точок, але мінімізація цих відстаней сама по собі призводить до вироджених сіток. Для якісного прогнозування сітки потрібні додаткові регуляризатори форми.

$$L_{edge}(V,E) = \frac{1}{|E|} \sum_{(v,v')\in E} ||v-v'||^2, \quad (2.18)$$

де $E \subseteq V \times V$ – ребра прогнозованої сітки.

Альтернативно, лапласівська втрата також накладає обмеження на гладкість. Втрата сітки на і-му етапі є зваженою сумою цих втрат. Гілка уточнення сітки навчається мінімізувати середнє значення цих втрат на всіх етапах уточнення. Більш детальна архітектура Mesh R-CNN представлена на рисунку 2.9, де можна побачити як дані після RoIAlign надходять до гілок і після кубіфікації створюється кубіфакована модель, потім створюється перша сітка і після аналогічної обробки фінальна сітка.

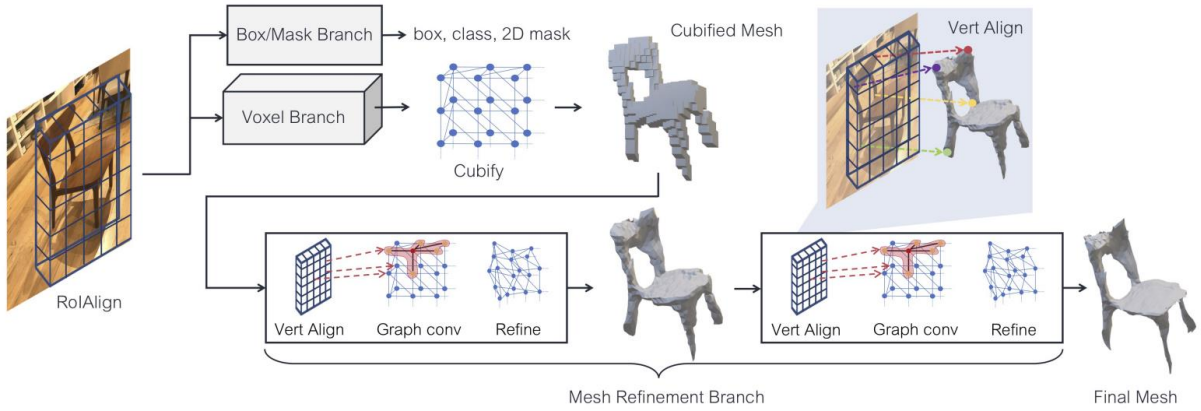

Рисунок 2.9 – Архітектура Mesh R-CNN

2.5 Архітектура GAN

Генеративна змагальна мережа (GAN) – це архітектура глибокого навчання, яка складається з двох нейронних мереж, які конкурують одна з одною в системі гри з нульовою сумою. Метою GAN є створення нових, синтетичних даних, які нагадують деякий відомий розподіл даних [18].

Генеративні змагальні мережі (GAN) можна розбити на три частини:

– генеративність: вивчення генеративної моделі, яка описує, як дані генеруються в термінах імовірнісної моделі;

– змагання: навчання моделі відбувається в умовах змагання;

– мережі: використання глибоких нейронних мереж як алгоритми штучного інтелекту для навчання.

Архітектура GAN складається з генератора і дискримінатора. Генератор створює підроблені зразки даних (зображення, аудіо тощо) і намагається довести Дискримінатору справжність вихідних даних. Дискримінатор, зі свого боку, намагається відрізнити справжні зразки від підроблених. Генератор і Дискримінатор є нейронними мережами, і обидва конкурують один з одним на етапі навчання. Ці кроки повторюються кілька разів, і після кожного повторення Генератор і Дискримінатор стають все кращими і кращими. Роботу можна наочно побачити на рисунку 2.10.

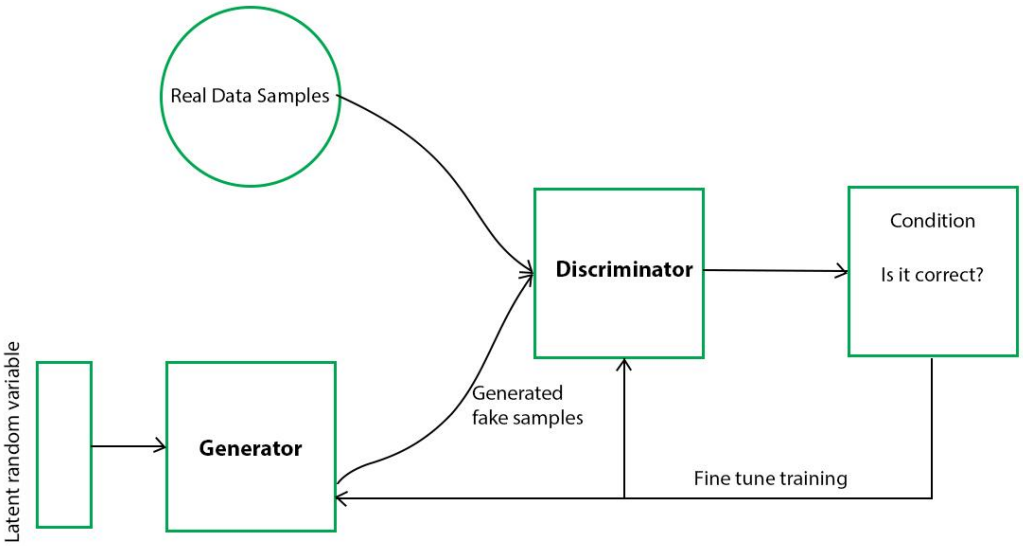

Рисунок 2.10 – Архітектура GAN

Генеративна модель фіксує розподіл даних і навчена таким чином, щоб вона намагалася максимізувати ймовірність помилки Дискримінатора. Дискримінатор, з іншого боку, заснований на моделі, яка оцінює ймовірність того, що отриманий зразок отримано з навчальних даних, а не з генератора. GAN сформульовано як мінімаксну гру, де Дискримінатор намагається мінімізувати свою винагороду V(D, G), а Генератор намагається мінімізувати винагороду Дискримінатора або, іншими словами, максимізувати свої втрати. Це можна математично описати наступною формулою:

$$\min_G \max_D V(D,G), \tag{2.19}$$

$$V(D,G) = E_{x\sim p_{data}(x)}[logD(x)] + \\ +E_{z\sim p_{data}(z)}[\log(1-D(G(z))], \tag{2.20}$$

де G – генератор;

D – дискримінатор;

$p_{data}(x)$, – розподіл реальних даних;

$p(z)$ – розподіл генератора;

x – вибірка з $p_{data}(z)$;

$p_{data}(z)$ – вибірка з $P(z)$;

$D(x)$ – мережа дискримінатора;

$G(z)$ – Генераторна мережа.

Отже, навчання GAN складається з двох частин:

– дискримінатор навчається, поки генератор не працює. На цьому етапі мережа передається лише в прямому напрямку, а зворотне не відбувається. Дискримінатор навчається на реальних даних для n епох і перевіряє, чи зможе він правильно передбачити їх як реальні. Крім того, на цьому етапі Дискримінатор також навчається на підроблених даних, згенерованих Генератором, і перевіряє, чи зможе він правильно передбачити їх як підроблені;

– генератор навчається, поки Дискримінатор не працює. Після того, як Дискримінатор навчено створеними фальшивими даними Генератора, ми можемо отримати його прогнози та використати результати для навчання Генератора та отримати кращий результат від попереднього стану, щоб спробувати збентежити Дискримінатор.

Vanilla GAN – це найпростіший тип GAN. Тут Генератор і Дискримінатор є простими багатошаровими перцептронами. У Vanilla GAN алгоритм дуже простий, він намагається оптимізувати математичне рівняння за допомогою стохастичного градієнтного спуску. Архітектуру Vanilla GAN представлено на рисунку 2.11.

Умовний GAN (CGAN) – CGAN можна описати як метод глибокого навчання, у якому встановлюються деякі умовні параметри [19]. У CGAN до Генератора додається додатковий параметр для генерації відповідних даних. Мітки також додаються до входу дискримінатора, щоб дискримінатор міг відрізнити справжні дані від фальшивих. Архітектуру

CGAN представлено на рисунку 2.12.

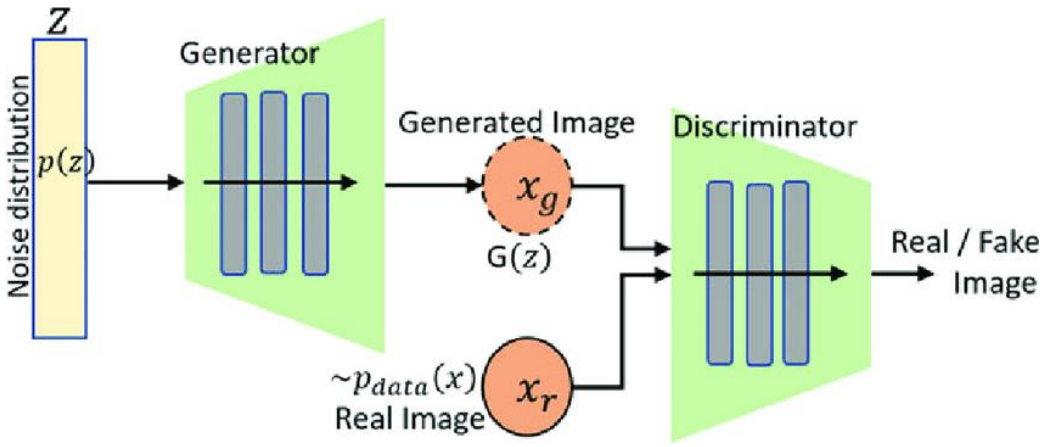

Рисунок 2.11 – Архітектура Vanilla GAN

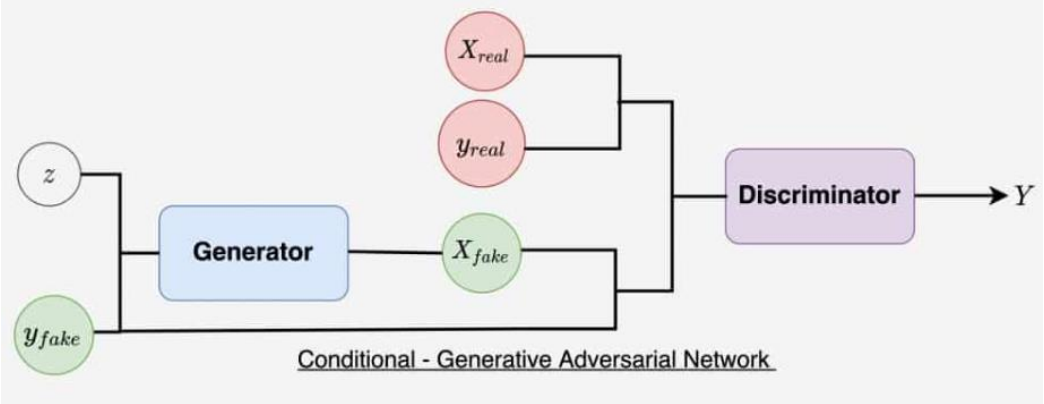

Рисунок 2.12 – Архітектура CGAN

GAN глибокої згортки – DCGAN є однією з найпопулярніших і найуспішніших реалізацій GAN [20]. Він складається з ConvNets замість багаторівневих перцептронів. ConvNets реалізовано без максимального об’єднання, яке фактично замінено згортковим кроком. Крім того, шари з’єднані не повністю. Архітектуру DCGAN представлено на рисунку 2.13.

Піраміда Лапласа GAN (LAPGAN) – ця модифікація архітектури використовує декілька мереж генератора та дискримінатора та різні рівні пірамід Лапласа [21]. Цей підхід використовується через змогу створення

дуже якісних зображень. Зображення спочатку зменшується на кожному шарі піраміди, а потім воно знову масштабується на кожному шарі у зворотному проході, де зображення отримує деякий шум від умовного GAN на цих шарах, доки воно не досягне свого початкового розміру. Архітектуру LAPGAN представлено на рисунку 2.14.

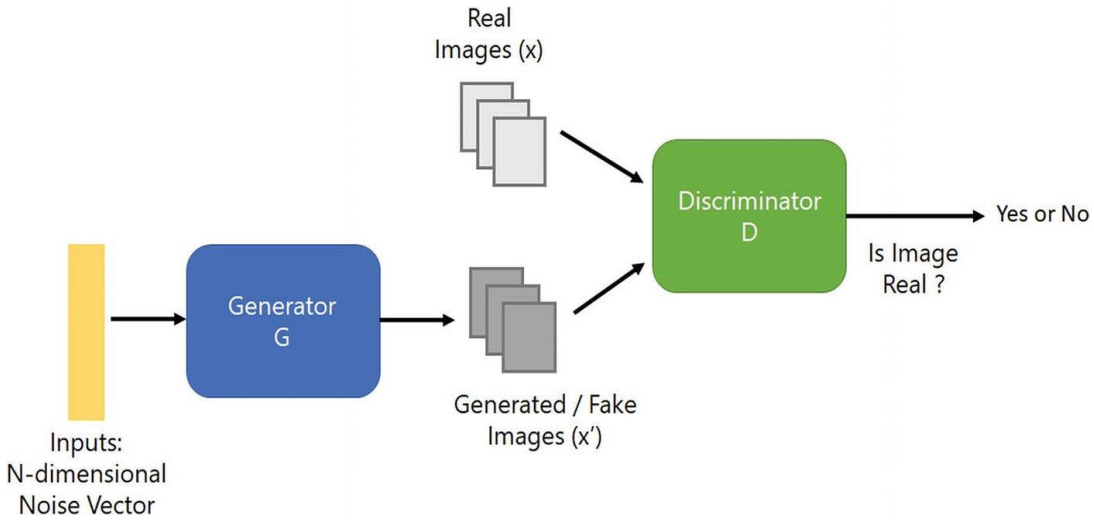

Рисунок 2.13 – Архітектура DCGAN

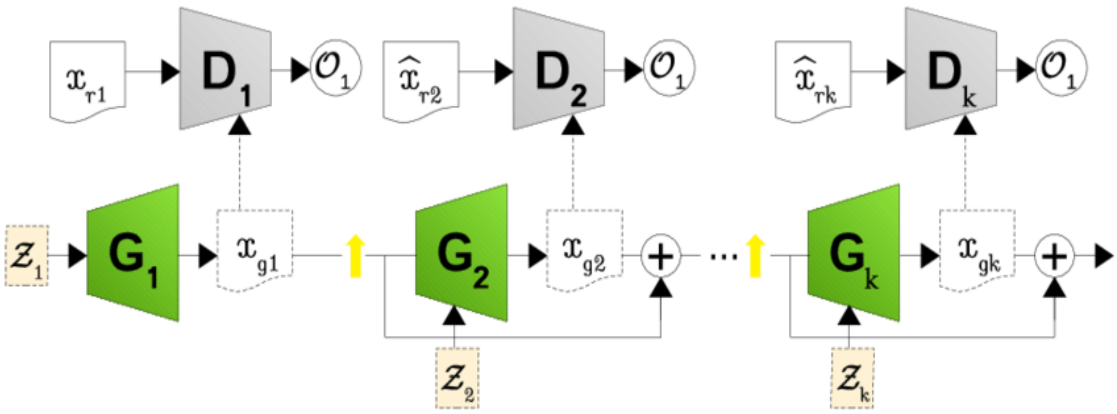

Рисунок 2.14 – Архітектура LAPGAN

GAN з надвисокою резолюцією: SRGAN – це спосіб розробки GAN, у якому глибока нейронна мережа використовується разом із суперницькою

мережею для створення зображень з вищою резолюцією. Цей тип GAN особливо корисний для оптимального масштабування оригінальних зображень із низькою резолюцією, щоб покращити їх деталі, мінімізуючи при цьому помилки. Архітектуру SRGAN представлено на рисунку 2.15.

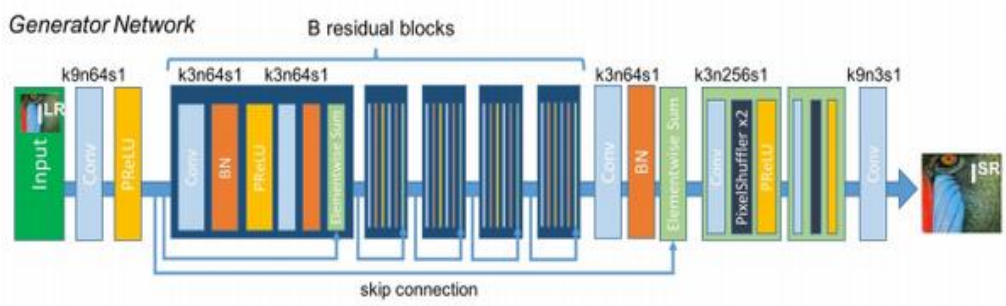

Рисунок 2.15 – Архітектура SRGAN

Для пропускної здібності шарів може бути використана індексація або як з $l-2$ до l або як з l до $l\ +\ 2$. Ці системи індексації є досить зручними при описуванні пропусків які йдуть у двох напрямках. Оскільки сигнал проходить напрямком вперед, простіше описувати пропуск як *к* із заданого рівня, але як правило під час зворотного поширення простіше описувати, який шар збудження буде використовуватись знову, як $l\ -\ k$, де $k\ -\ 1$ є числом пропуску.

Для заданої вагової матриці $W^{l-1,l}$ для ваг з'єднань з шару $l-1$ до l та вагової матриці $W^{l-2,l}$ для ваг з'єднань з шару $l-2$ до l прямим поширенням через передавальну функцію буде:

$$a^l := g(W^{l-1,l} * a^{l-1} + b^l + W^{l-2,l} * a^{l-2}) := \\ := g(Z^l + W^{l-2,l} * a^{l-2}), \tag{2.21}$$

де a^l – збудження (виходи) нейронів у шарі l;

g – передавальна функція шару l;

$W^{l-1,l}$ – вагова матриця для нейронів між шарами $l-1$ та l;

$Z^l = W^{l-1,l} * a^{l-1} + b^l$.

За відсутності явної матриці $W^{l-2,l}$ пряме поширення крізь передавальну функцію можна записати як

$$a^l := g(Z^l + a^{l-2}). \tag{2.22}$$

Інший спосіб сформулювати це – використовувати одиничну матрицю замість $W^{l-2,l}$, але це є справедливим лише коли розміри збігаються.

$$a^l := g(Z^l + \sum_{k=2}^{K} W^{l-k,l} * a^{l-k}. \tag{2.23}$$

Під час навчання зворотним поширенням для нормального шляху:

$$\Delta w^{l-1,l} := -\eta \frac{\partial E^l}{\partial w^{l-1,l}} = -\eta a^{l-1} * \delta^l. \tag{2.24}$$

І для шляхів пропусків (майже ідентично):

$$\Delta w^{l-2,l} := -\eta \frac{\partial E^l}{\partial w^{l-2,l}} = -\eta a^{l-2} * \delta^l, \tag{2.25}$$

де η – темпом навчання ($\eta < 0$);

δ^l – сигнал похибки нейронів на шарі l ;

a_i^l – збудження нейронів на шарі.

Якщо пропускний шаг має незмінні ваги (наприклад, одиничну матрицю, як вище), то вони не обчислюються. Якщо їх можливо обчислювати, то це правило є звичайним правилом обчислювання

поширення у зворотному напрямку. У загальному випадку може бути *K* вагових матриць шляхів пропуску, тож

$$\Delta w^{l-k,l} := -\eta \frac{\partial E^l}{\partial w^{l-k,l}} = -\eta a^{l-k} * \delta^l. \quad (2.26)$$

Оскільки навчання проходить однаково, вагові матриці можливо об’єднувати та навчати за один крок.

2.6 Залишкова нейронна мережа

Залишкова нейронна мережа (ResNet) насамперед вирішує проблему зникання градієнтів через використання концепції залишкових блоків [22]. Архітектура представлена на рисунку 2.16. У цій мережі використовується техніка під назвою пропуск штучних з’єднань. Пропускові з’єднання об’єднують активації шару з наступними шарами, пропускаючи деякі шари між ними. Це утворює залишковий блок. Залишкова мережа складається із залишкових одиниць або блоків, які мають пропускові з’єднання, які також називаються з’єднаннями ідентифікації.

Результати попереднього шару додаються до результатів шару після нього в залишковому блоці. Крок пропуску може бути 1, 2 або 3 шари. Під час додавання розміри x можуть відрізнятися від F(x) через процес згортання, що призводить до зменшення його розмірів. Таким чином, ми додаємо додатковий шар згортки 1 x 1, щоб змінити розміри x.

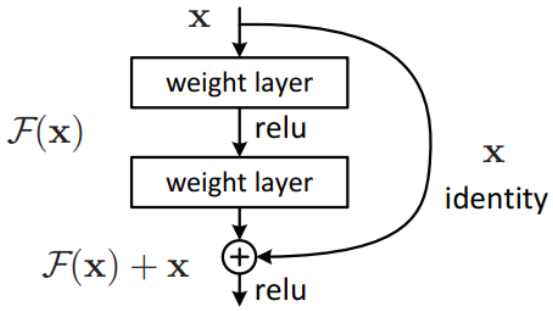

Рисунок 2.16 – Особливість архітектури ResNet

Перевага додавання цього типу пропуску з'єднання полягає в тому, що якщо будь-який шар зашкодить продуктивності архітектури, він буде пропущений регуляризацією. Таким чином, це призводить до навчання дуже нейронної мережі без проблем, викликаних зникненням градієнта оскільки існує менше шарів для поширення.

2.7 Graph Convolutional Network

Згорткова операція в Graph Convolutional Network – це, по суті, та сама операція як і згорткова операція в звичайній Convolutional Neural Network. Вона полягає у множенні вхідних нейронів на набір вагових коефіцієнтів, які зазвичай називають фільтрами або ядрами. Фільтри діють як ковзаюче вікно по всьому зображенню і дозволяють штучній нейронній мережі вивчати особливості сусідніх клітин. В межах одного шару один і той самий фільтр буде використовуватися на всьому зображенні. Наприклад, при використанні CNN для класифікації зображень котів і не-котів, той самий фільтр буде використано в тому самому шарі для виявлення носа і вух кота, що можна побачити на рисунку 2.17.

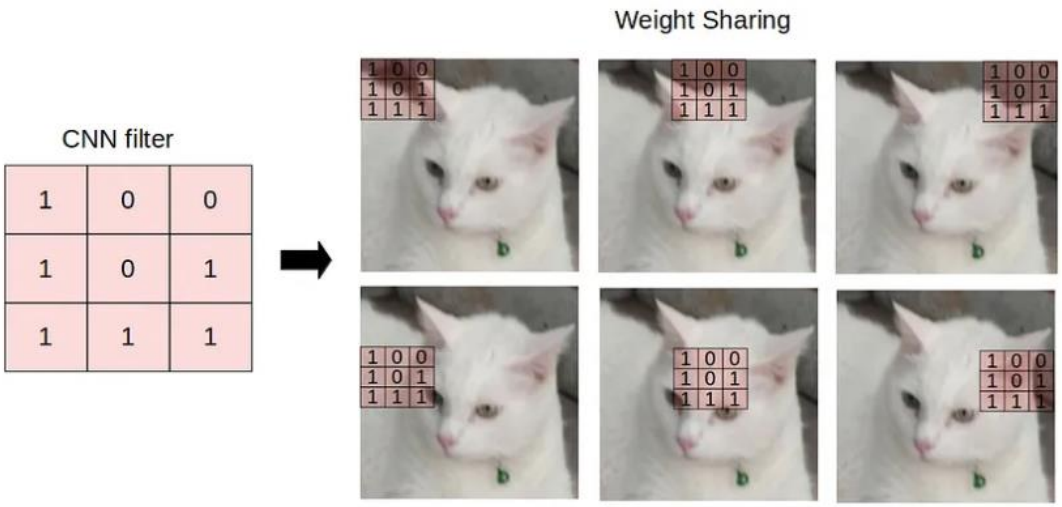

Рисунок 2.17 – Фільтр CNN

GCN виконують подібні операції, де модель вивчає особливості, оглядаючи сусідні вузли. Основна відмінність між штучної нейронної мережею і графовою нейронною мережею полягає в тому, що штучні нейронні мережі спеціально побудовані для роботи з регулярними, евклідовими, структурованими даними, тоді як графова нейронна мережа є узагальненою версією штучної нейронної мережі, де кількість зв’язків між вузлами варіюється, а вузли невпорядковані (нерегулярні на неевклідових структурованих даних). Схематичну різницю можна побачити на рисунку 2.18.

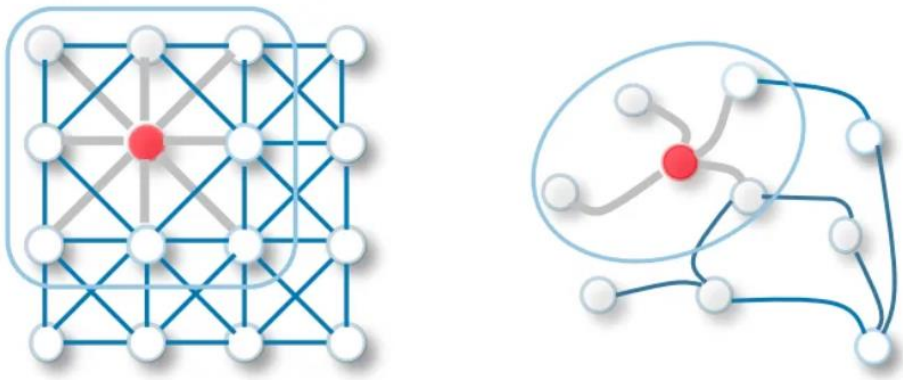

Рисунок 2.18 – CNN та GCN

GCN можна розділити на 2 великих алгоритми: згорткові мережі з просторовим графіком та згорткові мережі зі спектральним графіком, розглянемо алгоритм спектральної GCN.

Початкова ідея спектральної GCN була натхненна поширенням сигналу. Якщо уявити поширення інформації в спектральному GCN як поширення сигналу вздовж вузлів. Спектральні GCN використовують власне розкладання лапласової матриці графа для реалізації цього методу поширення інформації. Простіше кажучи, власний розклад допомагає нам зрозуміти структуру графа, а отже, класифікувати вершини графа. Це дещо схоже на базову концепцію аналізу головних компонент (PCA) та лінійного дискримінантного аналізу (LDA), де використовується власне розкладання для зменшення розмірності та виконання кластеризації.

У цьому підході враховується матриця суміжності A в рівнянні прямого поширення на додаток до характеристик вузлів (або так званих вхідних характеристик). A – це матриця, яка представляє ребра або зв’язки між вузлами в рівнянні прямого поширення. Додавання A в рівняння прямого проходження дозволяє моделі вивчати представлення ознак на основі зв’язку між вузлами. Заради простоти, зміщення b не враховується. Отриману GCN можна розглядати як наближення першого порядку до спектральної згортки графа у вигляді мережі передачі повідомлень, де інформація поширюється вздовж сусідніх вершин графа.

Додавши матрицю суміжності як додатковий елемент, рівняння прямого поширення матиме вигляд:

$$H^{i+1} = \sigma(W^i H^i A^*), \tag{2.27}$$

де A* – це нормалізована версія A.

2.8 Архітектура CGAN з використанням ResNet

Умовна GAN, що використовує одне зображення складається з генератора, який базується на Pixel2Mesh і дискримінатора, який використовує графову згорткову нейрону мережу (Graph Convolutional Network) для обробки нерегулярних об'єктів сітки.

Графова згорткова нейрона мережа це нейронна мережа, яка працює на графі. Дано меш M = (V, E, F), де V, E і F – вершини, ребра і векторів ознак, прикріплених до вершин мешу відповідно. Згортка графа для шару l визначається як:

$$f_p^{l+1} = w_0 f_p^l + \sum_{q \in N(p)} w_1 f_p^l, \quad (2.28)$$

де $f_p^{l+1} \in \mathbb{R}^{d_l}, f_p^{l+1} \in \mathbb{R}^{d_{l+1}}$ – вектори ознак на вершині p до і після згортки;

$N(p)$ є сусідні вершини p;

w_0 і w_1 – матриці навчальних параметрів $dl \times (dl + 1)$, які застосовуються до всіх вершини;

w_1 – спільним для всіх ребер, і, отже, працює на вузлах з різними ступенями вершин.

Процес згортки оновлює ознаки, що еквівалентно тому, що застосування деформації.

Для основи генератора можна використовувати мережу ResNet. Дискримінатор представляє собою семирівневу мережу, яка приймає згенерований об’єкт із генератора як вхідні дані та класифікує чи об'єкт справжній або підробка. Конкретніше, дискримінатор складається з трьох шарів згортки з розміром 16, 32 і 64 з подальшим проведенням max pooling і трьох лінійних шарів, які зводяться до одного значення. Також використовується сигмоїдна активація для визначення справжнього чи фальшивого передбачення.

Модель буде навчена за допомогою CGAN, яка генерує об’єкти за допомогою змагальної оцінки. Умовна GAN вивчає відображення з зображення x і вектор випадкового шуму z, до y, G: {x, z} → y. Генератор і

дискримінатор можна розглядати як двох гравців міні-макс гри, де генератор намагається «обдурити» дискримінатора, які спільно тренуються. Формалізувати мету моделі можна наступним чином:

$$l_{cGAN}(G,D) = E_{x,y}[logD(x,y)] + E_{x,z}[\log(1 - D(x,G(x,z))], \quad (2.29)$$

де G намагається мінімізувати цю мету проти D, який намагається максимізувати його:

$$G = arg\ minG\ maxD\ lcGAN\ (G,D). \quad (2.30)$$

Корисно змішувати геометричні втрати та регуляризації з метою GAN. Робота дискримінатора залишається незмінною, але генератор не тільки має збентежити дискримінатор, але й видавати геометрично подібні результати до оригінального об'єкта. Щоб визначати геометричну подібність, було визначено втрати, які можна застосувати до об'єктів. Була використовувана диференційна вибірка меша для вибірки хмар точок на поверхні об'єкту. Дано дві хмари точок P, Q з нормальними векторами. Нехай $A_{P,Q} = \{(p, argmin_q||p-q||) : p \in P\}$ – множина пар $(p,\ q)$ таких, що q є найближчим сусідом p в Q, а U_p – одинична нормаль до точки p. Фаска відстань між хмарами точок P і Q визначається за формулою 2.31.

$$l_{cham}(P,Q) = |P|^{-1} \sum_{(p,q)\in A_{P,Q}} ||p-q||^2 + |Q|^{-1} + \\ + \sum_{(q,p)\in A_{Q,P}} ||q-p||^2. \quad (2.31)$$

І абсолютна нормальна відстань визначається за формулою 2.32.

$$l_{norm}(P,Q) = -|P|^{-1} \sum_{(p,q)\in A_{P,Q}} |u_p * u_q| - |Q|^{-1} + \\ + \sum_{(q,p)\in A_{Q,P}} |u_q * u_p|. \quad (2.32)$$

Фаска та абсолютна відстані штрафують за невідповідність позицій та нормалей між двома хмарами точок, але мінімізація лише цих відстаней призводить до вироджених об'єктів (пересічні грані). Прогнозування високоякісних об'єктів потребує додаткові регуляризатори форми. Для цього були використані втрати ребер:

$$l_{edge}(V,E) = \frac{1}{|E|}\sum_{(v,v')\in E} ||v - v'||^2, \quad (2.33)$$

де $E \subseteq V \times V$ є ребрами прогнозованого об'єкта.

Загальна втрата об'єкта є зваженою сумою всіх чотирьох втрат:

$$l_{mesh} = l_{cham} + \lambda_1 l_{norm} + \lambda_2 l_{lap} + \lambda_3 l_{edge}, \quad (2.34)$$

де $\lambda_1 = 1.6e^{-4}$, $\lambda_2 = 0.3$, $\lambda_3 = 0.1$ є гіперпараметри, які врівноважують втрати та фіксовані для всіх експериментів.

Таким чином, кінцева мета може бути формалізована так:

$$G^* = arg \min_G \max_D l_{cGAN}(G,D) + \lambda l_{mesh}(G). \quad (2.35)$$

Архітектуру CGAN з використанням ResNet можна побачити на рисунку 2.19.

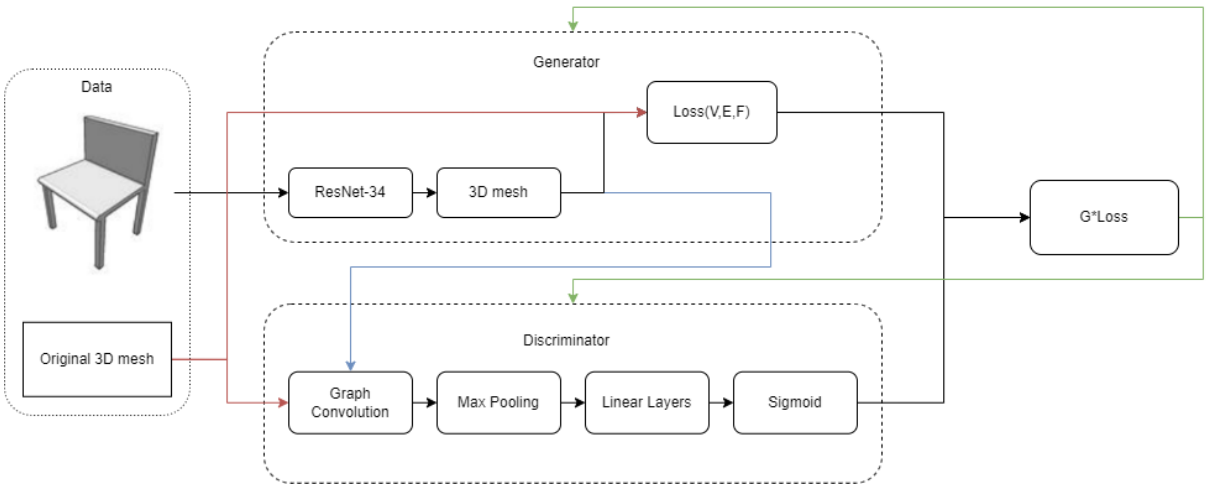

Рисунок 2.19 – Архітектура CGAN з використанням ResNet

3 ПРОВЕДЕННЯ ЕКСПЕРИМЕНТІВ

3.1 Опис набору даних

Набір даних складається з понад десяти тисяч 3D-сканів реальних об’єктів. Для створення набору даних було найнято 70 операторів. Їх забезпечили мобільними пристроями для 3D-сканування споживчого класу. Оператори сканували об’єкти на свій вибір, поза лабораторією і без безпосереднього нагляду фахівців з комп’ютерного зору. Результатом стала велика і різноманітна колекція сканів об’єктів: від взуття, чашок та іграшок до роялів, будівельної техніки та великих вуличних скульптур. Отримані дані були безповоротно розміщені в суспільному надбанні та доступні у вільному доступі.

Представлений набір даних містить понад 10 000 спеціальних 3D-сканів окремих об’єктів. Сканування були зроблені «в природних умовах» – в умовах, що імітують широке розгортання систем 3D-сканування серед споживачів. Оператори не були експертами в галузі комп’ютерного зору і не були безпосередньо знайомі з дослідженнями галузі комп'ютерного зору. Таким чином, сканування проводилося в умовах, які можуть бути схожими з тими, що виникають при використанні RGB-D камер споживчого класу для реконструкції об’єктів широким колом користувачів.

Широко розгорнуті системи реконструкції об’єктів, будуть базуватися на мобільних камерах споживчого класу, що експлуатуються поза контрольованим лабораторним середовищем користувачами без спеціальної підготовки чи досвіду. Оператори можуть тримати камеру або об’єкт в руках і вільно переміщати їх під час сканування. Набір даних був зібраний саме в такому режимі і може допомогти зрозуміти виклики, з якими стикаються під час реконструкції пайплайнів в умовах широкого розгортання.

Статистику набору даних та меші реконструкції можна побачити на рисунках 3.1 та 3.2.

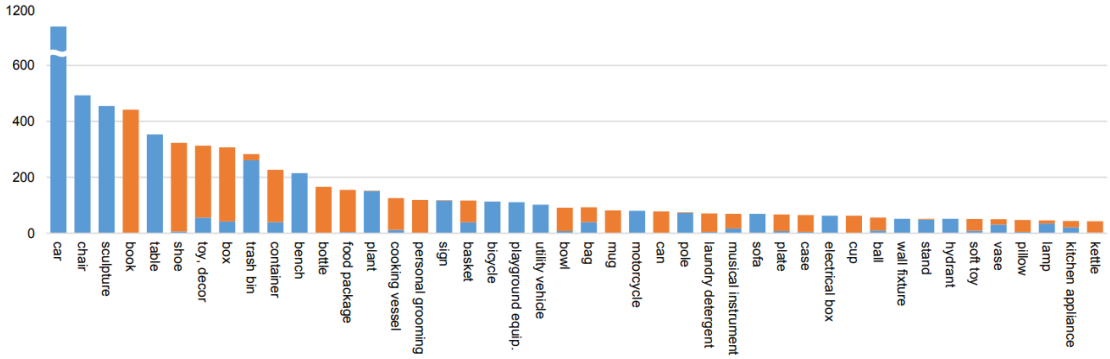

Рисунок 3.1 – Кількість сканувань у найбільших категоріях

Рисунок 3.2 – Різноманітність об'єктів, реконструйованих з 3D-сканів

Також для проведення тестів було обрано датасет ShapeNet. ShapeNet – це великомасштабне сховище 3D CAD-моделей, розроблене дослідниками зі Стенфордського, Прінстонського університетів та

Технологічного інституту Toyota в Чикаго. Репозиторій містить понад 300 мільйонів моделей, 220000 з яких класифіковані на 3135 класів, організованих за допомогою гіперо-гіпонімічних зв’язків WordNet. ShapeNet складається з декількох підмножин, однак, найбільш релевантною для сегментації є підмножина ShapeNet Parts. ShapeNet Parts містить 31693 сіток, розділених на 16 загальних класів об’єктів (наприклад, стіл, стілець, літак). Кожна фігура містить 2-5 частин (загалом 50 класів частин). Моделі з датасету можна побачити на рисунку 3.3.

Рисунок 3.3 – Моделі датасету ShapeNet CAD

3.2. Опис середовища розробки та обраної мови програмування

Для проведення експериментів було обрано мову програмування Python. Python – це високорівнева мова програмування з простим синтаксисом, що дозволяє розробникам швидко і легко створювати

програми. Python є інтерпретованою мовою, що означає, що програми пишуться у вигляді тексту і інтерпретуються рядок за рядком під час виконання. Також він має багато вбудованих бібліотек і модулів, що робить його дуже потужним і гнучким для розв’язання різноманітних задач.

Python широко використовується у сфері машинного навчання, оскільки має багато відкритих бібліотек та фреймворків, що дозволяють розробникам легко реалізувати моделі машинного навчання. У Python є потужні бібліотеки для машинного навчання, такі як TensorFlow, PyTorch, Keras, які дозволяють розробникам створювати складні моделі нейронних мереж і глибинного навчання. Ці бібліотеки також дозволяють використовувати різні алгоритми оптимізації, які забезпечують високу точність і швидкість виконання моделей.

У Python дуже зручно працювати з даними, які використовуються для машинного навчання. Бібліотеки, такі як Pandas і NumPy, дозволяють ефективно працювати з даними в форматі таблиць і масивів. Це дозволяє швидко підготувати дані для навчання моделей машинного навчання, а також обробляти, аналізувати та візуалізувати результати.

Python також дозволяє розробникам створювати власні модулі та пакети, що полегшує роботу з кодом і сприяє його повторному використанню. Крім того, Python має велику спільноту розробників, яка постійно розвиває і підтримує різноманітні бібліотеки та інструменти для машинного навчання. Він являється однією з найпопулярніших мов програмування для машинного навчання і штучного інтелекту в цілому. Використання Python у моделях машинного навчання дозволяє розробникам швидко і легко створювати складні моделі з високою точністю.

Для розробки програмного застосунку було обрано середовище PyCharm. PyCharm є інтегрованим середовищем розробки (IDE) для мови програмування Python. Воно було створене компанією JetBrains і є одним з найбільш популярних інструментів для розробки на Python.

Середовище має багато функцій, що полегшують роботу з Python,

таких як автодоповнення коду, рефакторинг, налагоджування, візуальне моделювання та підтримка віртуальних середовищ. Інтерфейс PyCharm простий і зрозумілий, але в той же час має багато можливостей для налаштування і розширення.

PyCharm має інтеграцію з багатьма іншими інструментами, такими як системи контролю версій (Git, Mercurial, SVN), бази даних, сервіси хмарного сховища та інші. Це дозволяє розробникам зручно і ефективно працювати з різними інструментами у своєму проекті. Останні версії PyCharm добре підтримують «ноутбуки», які часто використовують у візуалізації розподілів даних та отриманих результатів моделей.

Узагальнюючи, PyCharm – це потужне та зручне середовище розробки Python, яке дозволяє розробникам ефективно працювати з кодом, фреймворками, інструментами та різними сервісами, що робить його незамінним інструментом для розробників Python-програм.

3.3 Проведення та аналіз експерименту

Для реалізації генератора було повторно реалізовано мережу Pixel2Mesh на основі Mesh R-CNN. У цій реалізації було додано ResNet-50 замість архітектури VGG-16 для об’єднання перцептивних функцій. Ця реалізація перевершує оригінальну модель завдяки глибшій основі, кращому рецепту навчання та мінімізації фаски на вибірках, а не на позиціях вершин.

В якості дискримінатора було реалізовано неглибоку мережу з 7 шарів на основі графових згорткових шарів, яка приймає згенерований об’єкт з генератора на вхід і класифікує, чи є він справжнім (1) або ні (0), схему можна побачити на рисунку 3.4.

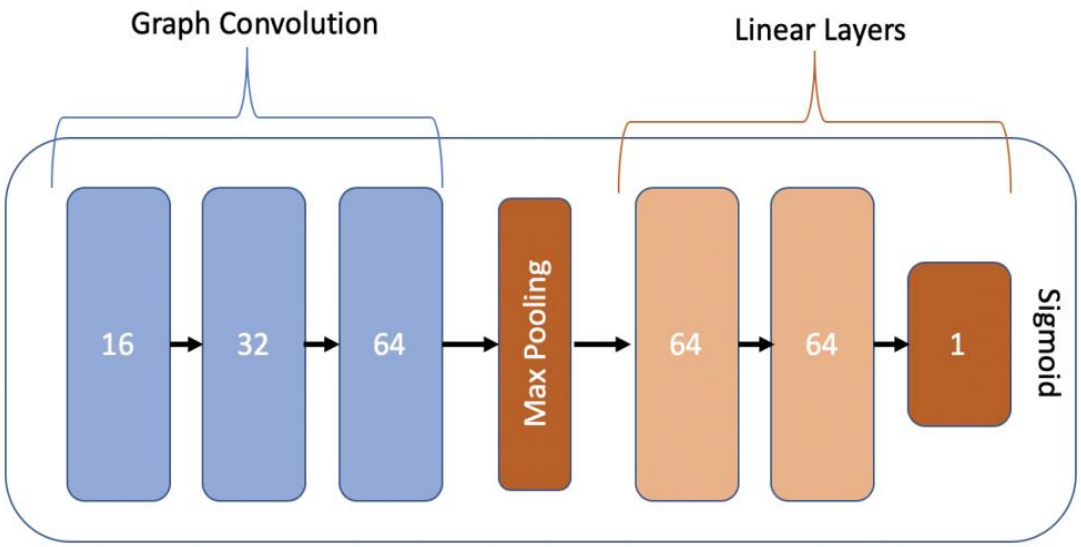

Рисунок 3.4 – Схематичне зображення дискримінатора

Для оптимізації мережі було застосовано наступний підхід: ми чергуємо один крок градієнтного спуску на дискримінаторі, а потім один крок на генераторі. Замість того, щоб тренувати генератор на мінімізацію log(1 - D(x, G(x, z)), тренуємо на максимізацію log D(x, G(x, z)). Також було використано мінібатч SGD та застосовано оптимізатор Адама зі швидкістю навчання 0.005, та параметрами імпульсу $\beta 1 = 0.45$, $\beta 2 = 0.7$.

Ми порівнюємо наш підхід з Pixel2Mesh, який прогнозує сітки шляхом деформації та розбиття початкового еліпсоїда. Ми врахували зміни, внесені в оригінальну архітектуру Pixel2Mesh кодом MeshRCNN для підвищення обчислювальної ефективності та надійного вилучення особливостей.

Ми досліджуємо вплив використання умовно-генеративної архітектури на одновимірну 3D-реконструкцію, реалізуючи наступні моделі.

У моделі 10k точок рівномірно відбираються випадковим чином з поверхні прогнозованої та істинної сіток і використовуються для обчислення відстані Chamfer. Ми також обчислюємо оцінку F1, використовуючи різні порогової відстані τ . Середнє гармонійне значення

відсотка прогнозованих точок в межах τ від точки істини дає точність, а навпаки – похибку. Для F1-рахунку, чим вище, тим краще, а для Chamfer distance, чим менше, тим краще.

З таблиці 3.1 можна побачити, що CGAN працює краще, ніж базова модель Pixel2Mesh як за показником F-рахунку, так і за показником відстані до фаски. Метрики оцінювання для генерації 3D-форм можуть не повністю відображати якість форми. Метрики часто фіксують заповненість або відстань між точками а не властивості поверхні, такі як неперервність, гладкість, деталізація високого порядку.

Таблиця 3.1 – Порівняння CGAN і Pixel2Mesh

Об’єкт	Модель	F1	Chamfer
Стілець	Pixel2Mesh	53.54	0.00061
	CGAN	55.67	0.00060
Стіл	Pixel2Mesh	62.49	0.00057
	CGAN	66.13	0.00055

На рисунку 3.5 можна побачити, що сітки, створені за допомогою CGAN, виглядають більш реалістичними, ніж ті, що згенеровані за допомогою Pixel2Mesh. Але також можна зазначити, що змодельовані об’єкти могли бути більш згладженої та натуральною форми ніж видають моделі. Також має сенс зауважити, що моделі працюють з вже виділеними об’єктами, що звісно являється зображенням, але краще якщо моделі зможуть видавати такий і кращий результат на кольорових фотографіях об’єктів.

Рисунок 3.5 – Порівняння результатів CGAN і Pixel2Mesh

Отже, можна затвердити, що модель з використанням ResNet і GCN виявила себе трохи краще, ніж мережа Pixel2Mesh. Але має сенс у покращенні моделі, тому що результат вийшов не ідеальний і працює лише з сегментованими зображеннями.

ВИСНОВКИ

У результаті дослідження, проведеного під час виконання кваліфікаційної роботи за темою «Глибинна нейронна мережа для створення 3D моделей на основі зображень об’єкта» було доведено актуальність перетворення зображень до 3D моделей. Було проведено ознайомлення з фотограмметрією, напрямками, що вивчають реконструкцію та генерацію 3D зображень. Здійснено аналіз існуючих рішень, саме платформ та сервісів, їх переваги і недоліки.

Проведено аналіз архітектур генеративно змагальних мереж, а саме Vanilla GAN, CGAN, LAPGAN, DCGAN, SarGAN і аналіз мереж ResNet і Graph Convolutional Network, що використовувались у розробці подальшій моделі CGAN. Також з архітектур було досліджено існуючи архітектури для вирішення поставленої задачі – 3D-R2N2, Mesh-RCNN та Pixel2Mesh. Pixel2Mesh використовувалась у створенні CGAN.

Після проведення дослідження було реалізовано Pixel2Mesh та CGAN архітектури на мові програмування Python з використанням бібліотек програмування нейронних мереж tensorflow та pythorch. Після навчання було проведено порівняння їх результатів на наборах даних redwood та ShapeNet CAD.

CGAN показала трохи кращий результат ніж Pixel2Mesh, модель змогла побудувати 3D модель з зображення стільця і стола, але є і певні недоліки – модель краще працюю з зображеннями вже після сегментації, на відміну від Mesh-RCNN, яка здібна будувати 3D модель по фотографії реальних об’єктів.

У майбутньому має певний сенс покращення моделі, або створення нового рішення для генерації 3D моделей з реальних фотографій об’єктів з мінімальною кількістю дій для цього – зробити одно фото об’єкта при майже любому освітленні для отримання якісної 3D моделей можна вважати успіхом у цьому напрямку.

Отже, під час виконання даної кваліфікаційної роботи, були удосконалені навички роботи з моделями машинного навчання. Було проведено дослідження у області фотограмметрії і генерації 3D моделей з 2D зображень. Проведено ознайомлення з архітектурами для вирішення поставленої задачи, після чого була удосконалена архітектура Pixel2Mesh. Проведено практичне дослідження з порівнянням результатів CGAN і Pixel2Mesh на наборах даних redwood і ShapeNet CAD, де CGAN виявила себе кращою ніж Pixel2Mesh і були означені цілі для подальшого розвитку дослідженя у напрямку генерації 3D моделей з 2D зображень.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Haykin, Simon S. Neural networks : a comprehensive foundation, 1999.

2. Bhadeshia H. K. D. H. Neural Networks in Materials Science, 1999.

3. Дорожинський О. Л. Аналітична та цифрова фотограмметрія: Навч. посіб. для студ. вузів; Нац. ун-т «Львів. політехніка», 2002.

4. Ina Jarve, Natalja Liba. The Effect of Various Principles of External Orientation on the Overall Triangulation Accuracy, 2010.

5. Baqersad Javad, Carr Jennifer. Dynamic characteristics of a wind turbine blade using 3D digital image correlation. Proceedings of SPIE, 2012.

6. NVIDIA's New Neural Network for Turning 2D Images into 3D Objects. URL: https://80.lv/articles/nvidia-s-new-neural-network-for-turning-2d-images-into-3d-objects/ (дата звернення: 08.04.2023).

7. NVIDIA Research Turns 2D Photos Into 3D Scenes in the Blink of an AI. URL: https://blogs.nvidia.com/blog/2022/03/25/instant-nerf-research-3d-ai/ (дата звернення 10.04.2023).

8. Jakub Langr, Vladimir Bok. GANs in Action: Deep learning with generative adversarial networks, 2019.

9. Alankrita Aggarwal, Mamta Mittal , Gopi Battineni. Generative adversarial network: An overview of theory and applications. URL: https://doi.org/10.1016/j.jjimei.2020.100004 (дата звернення: 14.04.2023).

10. Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition.

11. Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, Gabriele Monfardini. The Graph Neural Network Model. Volume: 20, Issue: 1, 2009.

12. Deep 2d-neural network and its fast learning / Y. Bodyanskiy et al. 2018 IEEE second international conference on data stream mining & processing (DSMP), Lviv, Ukraine, 21–25 August 2018. 2018. URL:

https://doi.org/10.1109/dsmp.2018.8478578 (дата звернення: 14.04.2023).

13. Bodyanskiy Y.,Boiko O., Pliss I., Kopaliani D., Volkova V.2D-Deep Neural Network and Its Online Rapid Learning. Proceedings of the 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), Metz, France, 2019, P. 304-307, 2019.

14. Bodyanskiy Y., Pliss I., Timofeev V. Discrete adaptive identification and extrapolation of two-dimensional fields. Pattern recognition and image analysis. 1995. Vol. 5, no. 3. P. 410-416.

15. Rock, J., Gupta, T., Thorsen, J., Gwak, J., Shin, D., Hoiem, D.: Completing 3d object shape from one depth image. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015.

16. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. ArXiv e-prints. 2015.

17. Abhijit Kundu, Yin Li, and James M. Rehg. 3d-rcnn: Instance-level 3d object reconstruction via render-andcompare. 2018.

18. Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. arXiv preprint arXiv:1812.02822, 2018.

19. Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. 2017.

20. Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. arXiv preprint arXiv:1812.07035, 2018.

21. Python3. URL: https://docs.python.org/3/ (дата звернення: 08.04.2023).

22. Рябова Н.В., Петрикін М.Ю. Штучні нейронні мережі в задачах створення 3D моделей на основі зображення об’єкта. Матеріали 27-го Міжнародного молодіжного форуму «Радіоелектроніка і молодь у XXI столітті». Україна, м. Харків, 10 – 12 травня 2023р.

ДОДАТОК А

Відомість кваліфікаційної роботи

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	63 с.
	Інші документи	
2.	Презентаційні матеріали	__ плакатів

.	.	Прізвище та ініціали.	Підп.	Дата			
Розробив		Петрикін М.Ю.			Глибинна нейронна мережа для створення 3D моделей на основі зображень об’єкта	Шифр групи	Код напр./спец.
Перевірив		Рябова Н.В.				СШІм-21-2	122
Н.контр.		Малєєва І.А.				ХНУРЕ кафедра ШІ	
Затв.		Філатов В.О.					